SOFTWARE METHODOLOGY

TABLE OF CONTENTS

APPENDICES

CHAPTER 1.

THIS DOCUMENT DEALS WITH THE "WHAT", "WHY", AND "HOW" OF PRODUCT SOFTWARE. IT HAS BEEN DEVELOPED TO SERVE AS A GUIDE FOR ALL ASPECTS OF PRODUCT SOFTWARE AND ITS SUPPORT FROM CONCEPT TO THE END OF ITS LIFE CYCLE. SEVERAL COMPLEX ISSUES ARE ADDRESSED TO MEET THE NEEDS OF GOOD SOFTWARE ENGINEERING DESIGN PROCTICE AND THE REQUIREMENTS OF RTCA/DO-178.

THE PRINCIPLES OF SOFTWARE ENGINEERING HAVE BEEN OBTAINED FROM SEVERAL LITERATURE SOURCES AND PAST EXPERIENCE. EVERYONE INVOLVED IN THE PRODUCTION OF SOFTWARE IN ENCOURAGED TO LOOK AT THE VARIOUS REFERENCES GIVEN IN THE BIBLIOGRAPHY AND STUDY SOFTWARE ENGINEERING CONCEPTS NOT ALREADY KNOWN. MOST OF THE PRACTICES CURRENTLY FAVORED HAVE EVOLVED OUT OF SOME VERY PAINFUL AND COSTLY EXPERIENCES OVER THE PAST TEN OR FIFTEEN YEARS. THROUGH GOOD SOFTWARE ENGINEERING PRACTICE, WE CAN LEARN FROM THIS EXPERIENCES WITHOUT REPEATING THEM.

THE METHODOLOGY IN THIS DOCUMENT HAS BEEN DESIGNED TO PRODUCE SOFTWARE THAT TOTALLY COMPLIES WITH RTCA DOCUMENT DO-178, "SOFTWARE CONSIDERATIONS IN AIRBORNE SYSTEMS AND EQUIPMENT CERTIFICATION", NOVEMBER, 1981. IT ALSO OUTLINES PROCEDURES FOR MAINTAINABILITY THAT ARE CONSISTENT WITH THE PRESENT DESIGN, QA, AND ECO PRACTICES NOW BEING USED BY KING RADIO FOR HARDWARE.

IN WRITING THIS METHODOLOGY DOCUMENT, WE HAVE TRIED TO KEEP SEVERAL GOALS IN MIND:

1. IT SHOULD BE ORGANIZED SO THAT IT IS EASY TO REVISE AND KEEP UP TO DATE. THIS IS ACCOMPLISHED BY GROUPING TOPICS AS MUCH AS POSSIBLE SO THAT THEY ARE COVERED IN NO MORE THAT ONE AREA.

2. TOPICS, CHAPTERS, AND APPENDICES ARE CHOSEN SO THAT A SPECIFIC AUDIENCE IS ADDRESSED. THIS WAY USERS HAVE TO READ ONLY THOSE SPECIFIC SECTIONS THAT APPLY TO THEIR TASK AND CAN IGNORE THE OTHERS.

3. DETAILED EXPLANATIONS OF PROCEDURES HAVE BEEN SEPARATED FROM EXPLANATIONS OF THE ESSENTIAL IDEAS. THIS ALLOWS USERS TO LEARN THE OVERALL CONCEPT WITHOUT READING UNINTERESTING DETAILS.

IT IS REASONABLE TO ASK WHAT BROAD TOPIC AREAS AND SPECIFIC ISSUES SHOULD BE ADDRESSED IN THE AREA OF SOFTWARE METHODOLOGY. THE FOLLOWING IS A LIST OF THESE SOFTWARE TOPICS AND ISSUES WITH A BRIEF EXPLANATION OF EACH:

1. SPECIFICATION - DESCRIBE WHAT THE SOFTWARE SHOULD DO IN ENOUGH DETAIL THAT A SOFTWARE DESIGN TEAM CAN READ THE WRITTEN SPECIFICATION AND DEVELOP A DETAILED SOFTWARE DESIGN PLAN.

2. DESIGN - ANALYZE THE SOFTWARE SPECIFICATIONS AND DETERMINE AN ARCHITECTURE FOR THE FUNCTIONS AND MODULES THAT WILL MEET THE SPECIFICATION AND BE TESTABLE TO THE LEVEL REQUIRED FOR CERTIFICATION.

3. DEVELOPMENT - TURN SOFTWARE DESIGN INTO CODE AND TEST MODULES FOR COMPLIANCE WITH THE DESIGN DESCRIPTION.

4. INTEGRATION - COMBINE HARDARE AND SOFTWARE DESIGNS INTO A WORKING PROCESSOR SYSTEM. WORK OUT HARDWARE/SOFTWARE COMPATIBILITY BUGS AND TEST THE SYSTEM TO SEE THAT IT MEETS BOTH HARDWARE AND SOFTWARE DESIGN REQUIREMENTS.

5. TEST - TESTING IS PERFORMED DURING DEVELOPMENT AND INTEGRATION TO VERIFY PERFORMANCE AGAINST REQUIREMENTS, MAINTAIN GOOD QUALITY, AND PROVIDE THE TEST DATA BASE NEED FOR CERTIFICATION.

6. QUALITY AUDIT - QUALITY ACCEPTANCE PROCEDURES MUST BE IMPLEMENTED DURING THE DESIGN AND DEVELOPMENT OF SOFTWARE BECAUSE THERE IS NO SOFTWARE "FACTORY". QUALITY AUDITING WOULD CONSIST OF CHECKING FOR COMPLIANCE WITH DESIGN STANDARDS AND CERTIFICATION REQUIREMENTS.

7. DOCUMENTATION - THE REQUIREMENTS, DESIGN, DEVELOPMENT, TESTING, AND RELEASE OF SOFTWARE MUST ALL BE DOCUMENTED TO PROVIDE GUIDANCE IN THE PRODUCTION OF QUALITY

SOFTWARE, TO INSURE THAT CERTIFICATION REQUIREMENTS ARE MET, AND TO PROVIDE FOR MAINTANENCE OF THE CODE OVER THE LIFETIME OF THE PRODUCT.

8. CERTIFICATION - THE CERTIFICATION OF SOFTWARE IS SUPPORTED BY DESIGN, DEVELOPMENT, TEST, AND SCM DOCUMENTATION. CERTIFICATION REQUIREMENTS ARE SPECIFIED IN RTCA DOCUMENT DO-178 AND VARY ACCORDING TO CRITICALITY CLASSIFICATION.

9. RELEASE - PROCESS USED TO GET COMPLETED SOFTWARE TO THE FACTORY. THE RELEASE PROCESS IS DESIGNED TO INSURE THAT ALL DOCUMENTS REQUIRED FOR CERTIFICATION ARE COMPLETED, THAT SOURCE CODE IS PROPERLY ARCHIVED, AND THAT MASTER CHIPS HAVE BEEN VERIFIED AND VALIDATED BY THE DESIGN TEAM PRIOR TO RELEASE TO THE FACTORY.

10. MAINTANENCE - MAINTANENCE OF SOFTWARE INVOLVES KEEPING SECURE ARCHIVED COPIES OF THE SOURCE CODE AND SUPPORT SOFTWARE AVAILABLE FOR FUTURE PRODUCT SOFTWARE REVISIONS AND MAKING ALL SOFTWARE CHANGES UNDER REVISION CONTROL.

11. PROJECT MANAGEMENT - PROMOTE GOOD QUALITY THROUGH FREQUENT DESIGN REVIEWS. ENCOURAGE EARLY PREPRATION OF DOCUMENTATION NEEDED TO SUPPORT DESIGN. EMPHASIZE IMPORTANCE OF TESTING AND DOCUMENTATION. ENCOURAGE USE OF SUPPORT TOOLS. IMPLEMENT A MOTHOD OF CONFIGURATION MANAGEMENT OF DEVELOPMENT SOFTWARE.

12 CONFIGURATION MANAGEMENT - DESIGN METHODS OF RELEASING AND ARCHIVING PRODUCT AND SUPPORT SOFTWARE THAT INSURE THAT THE EXACT CONFIGURATION FOR THE PRODUCT IS KNOWN. UTILIZE CHANGE CONTROL PROCEDURES THAT COMPLETELY DOCUMENT REVISIONS TO RELEASED SOFTWARE.

13. SUPPORT ENVIRONMENT - ALL ELEMENTS OF THE SOFTWARE DESIGN WORKING ENVIRONMENT THAT ARE NOT PRODUCT SOFTWARE. THESE INCLUDE THE VAX AND ITS EDITOR AND FILE MANAGER, ALL CROSS SOFTWARE ON VAX (COMPILERS, ASSEMBLERS, LOADERS, ETC.), EMULATORS, AND PROM PROGRAMMERS.

8888888888888888888888888888888888888888888888888888888888888888888888888888
STARTING NOTES:

SOFTWARE METHODOLOGY:

HOW DO YOU:

1) DESIGN IT
2) GET GOOD QUALITY
3) DOCUMENT IT
4) SPECIFY IT
5) TEST IT
6) CHANGE IT
7) MANAGE DESIGN
8) MANAGE CONFIGURATION
9) CERTIFY IT
10) RELEASE TO FACTORY

REQUIREMENTS FOR SOFTWARE	DOCUMENTATION
SOFTWARE DESIGN AND DEVELOPMENT	WRITE CODE REVIEW CODE TEST PLAN MANAGE FILES DOCUMENT DESIGN TEST MODULES ON HOST
SYSTEM INTEGRATION	EMULATION MODULE TEST FUNCTIONAL VERIFICATION OPERATIONAL VERIFICATION CODE REVISION TEST DOCUMENTATION
INITIAL RELEASE	SUBMIT CODE AND DOCUMENTS TO REVISION CONTROL. ISSUE AND VALIDATE MASTER CHIPS. ISSUE CHIPS TO MANUFACTURING. ARCHIVE VAX FILES.

SOFTWARE REVISION	DESIGNER GETS PREVIOUS CODE FROM ARCHIVES AND REVISES THE SOURCE. REPEAT STEPS OF INITIAL RELEASE.
QUALITY ASSURANCE	PROCEDURES DEFINED. PURPOSE OF IT. SCOPE. RESPONSIBILITY FOR IT. IMPLEMENTATION (FORMS).

SOFTWARE PROJECT MANAGEMENT

A PROJECT MANAGER FOR SOFTWARE MUST MANAGE PEOPLE AND THEIR COMMUNICATION AND DOCUMENTATION OF IDEAS. IT IS MORE DIFFICULT THAN HARDWARE PROJECT MANAGEMENT IN THE SENSE THAT THERE IS NO PHYSICAL EMBODIMENT. THE DESIGN EXISTS ONLY AS IDEAS AND COMPUTER CODE IN THE FORM OF WRITTEN DOCUMENTS. THE REVIEW AND MONITORING OF SOFTWARE DESIGN IS ALSO DIFFICULT BECAUSE PROGRESS REPORTING IS SO IMPRECISE. LITTLE CAN BE DONE EXCEPT TO REPORT ON THE NUMBER OF REQUIREMENTS THAT HAVE BEEN CODED AND TESTED.

THE BEST SOURCES OF CURRENT IDEAS ON SOFTWARE PROJECT MANAGEMENT ARE THE SOFTWARE ENGINEERING VIDEO TAPE COURSE FROM COLORADO STATE AND THE TEXT CALLED "SOFTWARE ENGINEERING" BY JENSEN AND TONIES. BOTH OF THESE REFERENCES ARE LISTED IN THE BIBLIOGRAPHY GIVEN IN THE APPENDIX.

EMPHASIS ON GOOD DOCUMENTATION AND DESIGN REVIEW TECHNIQUES IS THE KEY TO EFFECTIVE SOFTWARE PROJECT MANAGEMENT. PREPARATION OF DESIGN GUIDANCE DOCUMENTS EARLY IN THE PROJECT WILL GREATLY AID THE COMMUNICATION AND DESIGN REVIEW PROCESSES. DESIGN GUIDANCE DOCUMENTS ARE THE SOFTWARE REQUIREMENTS DOCUMENT, THE DESIGN DOCUMENT, AND THE TEST PLAN.

PREPARATION OF A DETAILED SOFTWARE REQUIREMENTS DOCUMENT IS ESSENTIAL IN PLANNING THE ARCHITECTURE AND BREAKING DOWN THE TASK INTO FUNCTIONS AND MODULES. THE TASK BREAKDOWN CAN THEN BE USED IN PREPARING SCHEDULES, DETERMINING CRITICAL PATHS, AND MAKING MANPOWER ASSIGNMENTS.

A "LIVING" DESIGN DOCUMENT IS ESSENTIAL FOR THE PROJECT TEAM TO USE AS A WORKING DEFINITION OF THE DESIGN. IT SHOULD DESCRIBE BOTH THE CHOSEN ARCHITECTURE AND THE PARTITIONING OF REQUIREMENTS INTO FUNCTIONS AND MODULES. A GOOD WORKING DEFINITION IS VALUABLE AS A COMMUNICATION TOOL TO INSURE THAT THE ENTIRE DESIGN TEAM IS WORKING TOWARD THE SAME GOALS. IT CAN ALSO BE EASILY MODIFIED TO ACCOMODATE NEW REQUIREMENTS AND NEW DESIGN APPROACHES.

TESTING REQUIREMENTS FOR CERTIFICATION NEED TO BE IDENTIFIED AND DOCUMENTED EARLY IN THE DESIGN PROCESS. TESTING FREQUENTLY IMPOSES MANY CONSTRAINTS ON DESIGN AND THESE NEED TO BE CONSIDERED BEFORE IT IS FINALIZED. THIS DOCUMENT WILL ALSO BE USED AS A WORKING DEFINITION OF THE TESTING APPROACH AND CAN BE MODIFIED AS NEEDED. THE DECISIONS ON WHAT TESTS NEED TO BE DONE ON THE VAX HOST AND WHAT TESTS NEED TO BE DONE ON THE EMULATOR SHOULD ALSO BE MADE EARLY. ACCEPTANCE TEST PROCEDURES CAN BE DEVELOPED LATER IN THE PROJECT BUT SHOULD NOT BE POSTPONED TOO LONG BECAUSE TEST FUNCTIONS MAY BE INCLUDED AS PART OF THE PRODUCT SOFTWARE AND NEED TO BE CONSIDERED IN DESIGN.

FREQUENT DESIGN REVIEWS NEED TO BE HELD AS AN AID TO ENHANCING COMMUNICATION ABOUT PROJECT REQUIREMENTS AND HOW THEY ARE BEING MET. DESIGN REVIEWS SHOULD NOT BE DESIGNING SESSIONS BUT DISCUSSIONS OF DESIGN APPROACHES AND PROBLEMS. THE ALSO AID THE PROJECT MANAGER IN MONITORING THE WORK PROGRESS IN EACH AREA. THE INFORMATION FROM DESIGN REVIEWS SHOULD BE USED TO UPDATE THE DESIGN GUIDANCE DOCUMENTS.

*888888

IT IS THE RESPONSIBILITY OF THE SOFTWARE PROJECT MANAGER TO SEE THAT THE CODE IS READY FOR RELEASE EARLY ENOUGH TO ALLOW TIME FOR THE RELEASE PROCESS TO PUT PROGRAMMED CHIPS INTO THE FACTORY.

SOFTWARE DEVELOPMENT SUPPORT ENVIRONMENT

THE SUPPORT ENVIRONMENT INCLUDES ALL OF THE VARIOUS TOOLS, FACILITIES, AND AIDS USED BY SOFTWARE DESIGNERS TO PRODUCE PRODUCT SOFTWARE AND GET IT INTO THE FACTORY. A PARTIAL LIST OF ELEMENTS OF THE SUPPORT ENVIRONMENT ARE:

1. VAX SERVICES
   - VMS OPERATING SYSTEM
   - UTILITIES
   - TEXT EDITOR
   - VAX FILE MANAGER
   - SYSTEM BACKUP

2. SUPPORT SOFTWARE ON VAX
   - COMMAND LINE INTERPRETER
     * COMPILERS
     * ASSEMBLERS
     * LOADERS
     * SIMULATORS
   - DEVELOPMENT CROSS SOFTWARE
     * COMPILERS
     * ASSEMBLERS
     * LINKING LOADERS
     * SIMULATORS
   - UTILITIES
     * ODD/EVEN BYTE SEPARATORS
     * CHIP MEMORY PARTITIONER
     * ASCII-HEX FORMATTER
     * INTERACTIVE HELP

3. SCM AND QA PROCEDURES
   - SCM FOR DEVELOPMENT SOFTWARE
   - QUALITY AUDIT
   - ACCEPTANCE TESTING
   - INITIAL RELEASE TO FACTORY
   - CONTROLLED DOCUMENTATION
   - ARCHIVE AND RETREVAL OF RELEASED SOFTWARE
   - MANAGEMENT OF VAX-BASED SUPPORT SOFTWARE

4. REAL-TIME EMULATION
   - SYMBOLIC DEBUG
   - MEMORY PARTITIONING
   - MODULE AND FUNCTION TESTING

5. PROM PROGRAMMING

[----------]

1. VAX SERVICES
   - VMS OPERATING SYSTEM
   - UTILITIES
   - TEXT EDITOR
   - VAX FILE MANAGER
   - SYSTEM BACKUP

[----------]

2. SUPPORT SOFTWARE ON VAX

SUPPORT SOFTWARE INCLUDES ALL VAX-BASED SOFTWARE NEEDED TO TRANSFORM TEXT FILES OF SOURCE CODE INTO FORMATTED ASCII-HEX TEXT FILES OF MACHINE CODE THAT CAN BE DOWNLOADED TO PROM PROGRAMMERS OR EMULATORS. IT ALSO INCLUDES DEVELOPMENT AIDS SUCH AS SIMULATORS AND INTERACTIVE HELP AS WELL AS AUTOMATIC DOCUMENTATION GENERATION. THIS SECTION WILL GIVE A BRIEF EXPLANATION OF THE VARIOUS SUPPORT SOFTWARE FUNCTIONS.

THE USE OF COMMAND INTERPRETERS TO RUN THE MICROPROCESSOR CROSS SOFTWARE IS PROBABLY ONE OF THE MOST INOVATIVE FEATURES OF OUR SUPPORT SOFTWARE. THEY APPEAR TO BE DCL COMMANDS FOR COMPILE, ASSEMBLE, LOAD, AND SIMULATE BUT ACTUALLY THEY ARE FOREIGN COMMAND PROCESSORS THAT HAVE BEEN DEVELOPED IN-HOUSE FOR THIS APPLICATION. WE CHOSE TO MAKE SUPPORT SOFTWARE COMMANDS APPEAR TO BE DCL SO THAT USERS

WOULD FEEL THAT THEY WERE IN THE FAMILIAR VMS ENVIRONMENT. WHEN A USER ENTERS ONE OF THESE COMMANDS, A SYSTEM-WIDE LOGICAL NAME ASSIGNMENT POINTS TO ONE OF THE EXECUTABLE ELEMENTS FOR THE SPECIFIC FOREIGN COMMAND PROCESSOR AND THE USER'S JOB IS CONTROLLED BY THE CUSTOM SOFTWARE.

WHEN A COMMAND SUCH AS $ ASSEMBLE KY196V01.I48/8048 IS ENTERED, THE COMMAND INTERPRETER READS THE FILE NAME AND CALLS THE 8048 ASSEMBLER TO ASSEMBLE THE FILE BY THAT NAME. THE COMMAND INTERPRETER DOES NOT CONTAIN THE ASSEMBLERS BUT MERELY CALLS THEM. SPECIFIC VERSIONS OF ALL COMMAND PROCESSORS ARE DEFINED IN A COMMAND PROCEDURE MAINTAINED BY SOFTWARE CONTROL. THE COMMAND INTERPRETER CAN BE THOUGHT OF AS A FRIENDLY INTERFACE THAT STAYS THE SAME AS THE SUPPORT SOFTWARE IS REVISED AND AVOIDS CONFUSION ABOUT WHICH VERSION OF SUPPORT SOFTWARE TO USE.

THE NEXT ITEM OF SUPPORT SOFTWARE IS THE VARIOUS PACKAGES OF DEVELOPMENT CROSS SOFTWARE SUCH AS COMPILERS, ASSEMBLERS, LINKING LOADERS, AND SIMULATORS. CURRENTLY THERE ARE ASSEMBLERS, LINKING LOADERS, AND SIMULATORS FOR ALL OF THE POPULAR MICROPROCESSORS AND COMPILERS FOR THE Z8002 AND 8086. ALL CROSS SOFTWARE IS ACCESSED THROUGH THE APPROPRIATE COMMAND PROCESSOR.

AT THE PRESENT TIME THERE IS NO STANDARD ORGANIZATION FOR THE COMPILERS. THEIR IMPLEMENTATION IS QUITE COMPLEX AND VARIES WIDELY FROM VENDOR TO VENDOR AND DEVICE TO DEVICE. IT IS OUR GOAL TO HOST COMPILERS WHICH PRODUCE A COMMON INTERMEDIATE MACRO CODE THAT CAN BE INTERPRETED TO PRODUCE MACHINE SPECIFIC ASSEMBLY CODE. THESE GENERAL INTERFACES WILL MAKE CHANGING PROCESSOR TYPES MUCH EASIER AND WILL ALLOW US TO INTERFACE THEM TO OUR PRESENT SUPPORT SOFTWARE.

WE HOST ASSEMBLERS THAT PRODUCE RELOCATABLE OBJECT FILES THAT ARE COMPATIBLE WITH THE LINKING LOADERS. THIS IS NECESSARY SO THAT SOFTWARE MODULES CAN BE SEPARATELY ASSEMBLED AND LINKED. THE 8048 IS AN ABSOLUTE ASSEMBLER AND CANNOT PRODUCE RELOCATABLE OBJECTS. THE COMMAND PROCESSOR FOR ASSEMBLERS ALLOWS THE USE OF THE QUALIFIER /ABS WHICH DIRECTS THE ASSEMBLER TO PRODUCE A LOAD FILE (A SINGLE ABSOLUTE OBJECT FILE). THIS IS DONE FOR THE CONVIENENCE OF SINGLE CHIP DESIGNS THAT HAVE ONLY ONE SOURCE FILE.

THE LOADER (OR LINKING LOADER) COMBINES SEVERAL RELOCATABLE OBJECT FILES INTO A SINGLE LOAD FILE (ABSOLUTE OBJECT FILE). IT LINKS THEM ALL TOGETHER AND RESOLVES ALL SYMBOLS AND ADDRESSES. IN SOME CASES A LIBRARIAN IS USED IN CONJUNCTION WITH THE LOADER TO ACCESS A COMMON LIBRARY TO RESOLVE SYMBOLS. A LIBRARIAN MAY OR MAY NOT BE PART OF THE LOADER.

SIMULATORS ARE PROVIDED TO ALLOW EXECUTION AND TESTING OF MACHINE CODE DIRECTLY ON VAX WITHOUT NEEDING EITHER AN EMULATOR OR THE TARGET HARDWARE. THE SIMULATOR HAS DEBUG CAPABILITY AND PROVIDES DATA ON MACHINE EXECUTION OF THE CODE INCLUDING TIMING INFORMATION IN MACHINE CYCLES AND STATUS OF ALL REGISTERS. THE SIMULATORS DO NOT EXECUTE IN REAL TIME AND SHOULD BE USED MAINLY FOR MODULE TESTING AND TIMING MEASUREMENTS.

THE NEXT GROUP OF SUPPORT SOFTWARE PACKAGES ARE REFERRED TO AS UTILITIES BECAUSE THEY ARE NOT DIRECTLY INVOLVED IN PRODUCING MACHINE CODE. THEY ARE USED FOR CONVIENENCE IN PRODUCING DOCUMENTATION OF RELEASED SOFTWARE AND FOR PRODUCING MACHINE CODE FILES THAT HAVE BEEN ORGANIZED PROPERLY FOR DIRECT DOWNLOAD INTO MASTER MEMORY CHIPS.

WHEN 8-BIT WIDE MEMORY IS USED FOR 16-BIT MICROPROCESSOR SYSTEMS, THE 16-BIT WIDE WORDS MUST BE SEPARATED INTO LOWER AND UPPER (OR ODD AND EVEN) BYTES SO MEMORY CAN BE LOADED. THE SEPARATOR DOES THIS AUTOMATICALLY AND PRODUCES TWO TEXT FILES. THE FIRST OR ODD FILE CONTAINS ALL OF THE LOWER BYTES AND THE SECOND OR EVEN FILE CONTAINS ALL OF THE UPPER BYTES. THESE ARE NO LONGER MACHINE EXECUTABLE FILES BUT MUST BE LOADED INTO APPROPRIATELY ORGANIZED MEMORY CHIP SETS TO BE EXECUTED.

THE PARTITIONER PERFORMS A VARITY OF TASKS. IT PARTITIONS THE LOAD FILE INTO MEMORY-CHIP SIZED SEGMENTS SO THE CODE CAN BE CONVIENENTLY DOWNLOADED INTO THE SPECIFIED ROM MEMORY. IT PRODUCES A STANDARD FORMATTED DISPLAY OF ALL MACHINE ADDRESSES AND ASCII-HEX CODE INSTRUCTIONS. IT LOADS ALL UNEXECUTED MEMORY LOCATIONS WITH THE DEFAULT HEX INSTRUCTION FF. AND FINALLY, IT REMOVES ALL FORMAT SPECIFIC INFORMATION SUCH AS DELIMITERS AND CHECK SUMS. THE PARTITIONER PRODUCES A TEXT FILE, CALLED THE MACHINE CODE DOCUMENT, (.DOC FILE EXTENSION) THAT IS A HUMAN READABLE AND LEGAL REPRESENTATION OF THE MACHINE CODE EXECUTED BY THE MICROPROCESSOR.

THE FORMATTER USES THE MACHINE CODE DOCUMENT FILE AS AN INPUT TO PRODUCE A DOWNLOADABLE ASCII-HEX FILE IN ONE OF SEVERAL SELECTABLE STANDARD FORMATS. A FORMATTED HEX FILE IS PRODUCED FOR EACH MEMORY DEVICE IN THE TARGET HARDWARE. FORMATTING AT THIS STAGE IS DONE FOR TWO REASONS. FIRST, A FORMATTED FILE (EXAMPLE WOULD BE AN INTEL HEX FORMAT) IS REQUIRED BY ALL EMULATORS AND PROM PROGRAMMERS FOR DOWNLOAD. SECOND, THE FORMATTER USES THE LEGAL REPRESENTATION OF THE MACHINE CODE THEREBY INSURING THAT WHAT IS LOADED IN MEMORY IS WHAT IS DOCUMENTED.

**888 HERE TELL ABVOUT HOW THE SYSTEM IS MUCH MORE COMPLEX THAN NEEDED FOR A SINGLE CHIP SYSTEM AND HOW THE SINGLE CHIP USER MAY BE FRUSTRATED BUT TELL HOW THE SYSTEM ALSO FITS THE SINGLE CHIPPER AND THE IMPORTANCE OF DOCUMENTATION.

IN A SINGLE-CHIP SYSTEM, THE ABSOLUTE OBJECT FILE COULD BE LOADED DIRECTLY INTO PROGRAMMBLE DEVICES; HOWEVER, IN MULTI-CHIP SYSTEMS THIS WOULD NOT BE POSSIBLE. FURTHERMORE, IT IS THE RESPONSIBILITY OF SOFTWARE CONTROL TO PROVIDE A LEGAL REPRESENTATION OF THE CONTENTS OF OUR PROGRAMMED EPROMS; AND SUCH DOCUMENTATION IS MADE MOST VALUABLE THROUGH THE USE OF A STANDARDIZED FORMAT OF DOWNLOADABLE OBJECT CODE. FOR THIS REASON, ALL LOAD FILES MUST BE PARTITIONED AND FORMATTED, WHETHER OR NOT THESE OPERATIONS WOULD BE REQUIRED BY THE NATURE OF THE SYSTEM.

[----------]

3. SCM AND QA PROCEDURES
   - SCM FOR DEVELOPMENT SOFTWARE
   - QUALITY AUDIT
   - ACCEPTANCE TESTING
   - INITIAL RELEASE TO FACTORY
   - CONTROLLED DOCUMENTATION
   - ARCHIVE AND RETREVAL OF RELEASED SOFTWARE
   - MANAGEMENT OF VAX-BASED SUPPORT SOFTWARE

[----------]

4. REAL-TIME EMULATION
   - SYMBOLIC DEBUG
   - MEMORY PARTITIONING
   - MODULE AND FUNCTION TESTING

[----------]

5. PROM PROGRAMMING

CERTIFICATION PER RTCA DO-178

INTRODUCTION

PRODUCING PRODUCT SOFTWARE THAT CAN BE CERTIFIED PER DO-178 INVOLVES THE FOLLOWING:

A. ACTIVITIES:

1. IDENTIFICATION OF LEVEL OF CRITICALITY.
2. APPLICATION OF GOOD DESIGN PRACTICES.
3. APPLICATION OF PROVEN SOFTWARE PROJECT MANAGEMENT TECHNIQUES.
4. PREPARATION OF PRODUCT SOFTWARE DOCUMENTATION.
5. FREQUENT DESIGN REVIEWS.
6. TESTING THAT DEMONSTRATES COMPLIANCE WITH REQUIREMENTS.
7. MODULE AND FUNCTIONAL TESTING TO REDUCE POTENTIAL BUGS TO A LOW LEVEL.
8. FIELD TESTING TO DEMONSTRATE OPERATIONAL CHARACTERISTICS AND COMPATIBILITY WITH FULL SYSTEM ENVIRONMENT.
9. CONFIGURATION MANAGEMENT (SCM) TO INSURE TRACABILITY.
10. QUALITY CONTROL (QA).
11. CERTIFICATION OF PRODUCT SOFTWARE DESIGN.
12. REVISION CONTROLLED RELEASE TO FACTORY.

B. DOCUMENTATION OF PRODUCT SOFTWARE DESIGN:

1. REQUIREMENTS
2. DESIGN DESCRIPTION.
3. CONFIGURATION INDEX.
4. SOURCE LISTING.
5. PROGRAMMER'S REFERENCE LITERATURE.
6. TEST PLANS/PROCEDURES THAT SUPPORT SPECIFIED LEVEL OF CRITICALITY.
7. TEST RESULTS SHOWING COMPLIANCE WITH BOTH CRITICALITY AND OPERATIONAL REQUIREMENTS.
8. SCM AND QA PLAN.

C. DOCUMENTATION OF SOFTWARE PROCEDURES, PRACTICES, AND STANDARDS.

1. DESIGN AND CODING PRACTICES AND STANDARDS.
2. CONFIGURATION MANAGEMENT AND QUALITY ASSURANCE PROCEDURES.
   - A) CONFIGURATION CONTROL.
   - B) CONFIGURATION IDENTIFICATION.
   - C) CONFIGURATION STATUS ACCOUNTING.
   - D) REVIEWS AND QUALITY AUDITS.
   - E) PROBLEM REPORTING AND CORRECTIVE ACTION PROCEDURE.
   - F) CHANGE CONTROL AND MAINTANENCE PROCEDURES.
3. DEVELOPMENT SUPPORT ENVIRONMENT.

THE CREATION OF HIGH QUALITY PRODUCT SOFTWARE THAT CAN BE CERTIFIED IS REALLY JUST A MATTER OF FOLLOWING THE CURRENTLY RECOMMENDED PRACTICES OF SOFTWARE ENGINEERING. THESE PRACTICES CAN BE SUMMARIZED AS FOLLOWS:

1. DOCUMENT A DETAILED SYSTEM DEFINITION (SYSTEM SPECIFICATION) OF THE PRODUCT PRIOR TO THE START OF ANY DESIGN EFFORT.

2. PREPARE A SOFTWARE REQUIREMENTS DOCUMENT (SOFTWARE SPECIFICATION) FROM THE SYSTEM DEFINITION. MAKE SURE EACH REQUIREMENT CAN BE TESTED. IDENTIFY THE LEVEL OF CRITICALITY SO APPROPRIATE TEST PLANS CAN BE DEVELOPED. CLEARY IDENTIFY THOSE SOFTWARE DOCUMENTS THAT HAVE TO BE DELIVERED AS PART OF THE CERTIFICATION ACTIVITY.

3. PREPARE A DETAILED DESIGN PLAN THAT MEETS ALL REQUIREMENTS AND THAT WILL LEAD TO DEVELOPMENT OF MODULES AND FUNCTIONS THAT CAN BE TESTED TO THE SPECIFIED LEVEL OF CRITICALITY.

4. PREPARE A TEST PLAN THAT SHOWS HOW THE SOFTWARE WILL BE TESTED TO MEET THE SPECIFIED LEVEL OF CRITICALITY. IT MUST BE DETAILED ENOUGH TO SPECIFY A PASS/FAIL CRITERION FOR EACH SOFTWARE REQUIREMENT.

5. PREPARE A DETAILED PROJECT SCHEDULE BEING SURE TO ALLOW ADEQUATE TIME FOR DETAILED TESTING, HARDWARE/SOFTWARE INTEGRATION, AND FREQUENT DESIGN REVIEWS (CODE WALKTHROUGHS,

REQUIREMENTS REVIEWS, RESULTS OF TESTING, ETC) IN ADDITION TO THE NORMAL CODE WRITING EFFORT.

6. CONDUCT FREQUENT ENOUGH DESIGN AND PROJECT STATUS REVIEWS THAT PROBLEMS ARE DETECTED BEFORE THEY CAN IMPACT DESIGN QUALITY.

7. DETERMINE WHAT PROJECT DOCUMENTATION WILL BE NEEDED TO COMPLY WITH CERTIFICATION REQUIREMENTS AND HAVE PROJECT PEOPLE WORK ON IT IN PARALLEL WITH THE DEVELOPMENT EFFORT. IT CAN PROVIDE VALUABLE INFORMATION FOR PROJECT MANAGEMENT PURPOSES AND WILL BE BETTER QUALITY THAN IF IT IS PUT OFF UNTIL THE END OF THE PROJECT. CONDUCT REVIEWS OF THE DOCUMENTATION ALONG WITH THE DESIGN.

8. DEVELOP AN SCM AND QA PLAN THAT SHOWS HOW THE PROJECT WILL BE MANAGED TO COMPLY WITH THE NECESSARY SOFTWARE CONFIGURATION MANAGEMENT AND QUALITY ASSURANCE PRACTICES AND PROCEDURES. BE SURE THAT ALL PROJECT PEOPLE KNOW WHAT THEY WILL BE REQUIRED TO DO TO SUPPORT SCM AND QA.

9. USE PROJECT-CONTROLLED CONFIGURATION MANAGEMENT OF DEVELOPMENT SOFTWARE SO PROJECT STATUS CAN BE TRACED AND TESTING RESULTS CAN BE RELATED TO A SPECIFIC PRODUCT SOFTWARE VERSION.

10. DETERMINE WHAT DESIGN AND CODING STANDARDS APPLY TO THE PROJECT AND MAKE SURE ALL PROJECT PEOPLE ARE AWARE OF THEM AND FOLLOW THEM.

11. MAKE SURE ALL PROJECT PEOPLE ARE FAMILIAR WITH THE MICROPROCESSOR DEVELOPMENT SUPPORT ENVIRONMENT AND ARE USING THE BEST TOOLS FOR THE APPLICATION.

12. PERFORM AND DOCUMENT FIELD TESTS TO DEMONSTRATE COMPLIANCE WITH EACH OPERATIONAL REQUIREMENT.

13. REVIEW PRODUCT SOFTWARE FINAL DOCUMENTATION TO SEE THAT IT SUPPORTS ALL CERTIFICATION REQUIREMENTS AND IS DETAILED ENOUGH ALLOW EASY MODIFICATION AND RETESTING. THIS IS ESPECIALLY IMPORTANT FOR THE DESIGN DESCRIPTION, COMMENTED SOURCE CODE, AND TEST PLANS/PROCEDURES.

14. PREPARE THE SOFTWARE DOCUMENT PACKAGE THAT WILL BE SUBMITTED FOR CERTIFICATION. BE SURE TO GET APPROVAL FOR RELEASE OF ANY DOCUMENTS THAT MAY BE CONSIDERED CONFIDENTIAL.

15. SCHEDULE THE FINAL SOFTWARE RELEASE WITH ENOUGH LEAD TIME THAT MASTER CHIPS CAN BE PROGRAMMED BY SOFTWARE CONTROL, VALIDATED BY DESIGN ENGINEERING, AND MASS PROGRAMMED BY RECEIVING INSPECTION IN QUANTITIES THAT WILL SUPPORT THE FACTORY.

SOFTWARE DOCUMENTATION

INTRODUCTION

SINCE SOFTWARE IS INFORMATION AND IDEAS AND NOT A PHYSICAL ENTITY, THE SOFTWARE "PRODUCT" IS CONSIDERED TO BE THE DOCUMENTATION PACKAGE THAT DESCRIBES IT. SOFTWARE SHOULD NOT BE THOUGHT OF AS PROGRAMMED CHIPS, OR MACHINE OP CODES, OR IMAGES ON MAGNETIC TAPE. THESE DO NOT COMMUNICATE INFORMATION OR IDEAS AND CANNOT BE USED TO MAINTAIN OR IMPROVE THE SOFTWARE "PRODUCT".

MOST SOFTWARE ENGINEERS AND CERTIFICATION ORGANIZATIONS WOULD JUDGE THE QUALITY OF A SOFTWARE PRODUCT BY THE QUALITY OF ITS DOCUMENTATION. IT IS WITH THIS DOCUMENTATION THAT DESIGN IDEAS AND TEST RESULTS MAY BE EVALUATED. GOOD DOCUMENTATION ALSO PROVIDES FOR EASIER MAINTANENCE AND MORE RELIABLE MODIFICATIONS.

POOR DOCUMENTATION OFTEN IMPLIES, RIGHTLY OR WRONGLY, A DESIGN THAT HAS BEEN POORLY CONCEIVED AND TESTED AND HENCE WILL BE UNRELIABLE AND DIFFICULT TO MAINTAIN. HOPEFULLY, THE CASE FOR A QUALITY DOCUMENTATION PACKAGE HAS BEEN ADEQUATELY MADE AND THE REST OF THIS DISCUSSION CAN FOCUS ON DETAILS.

SOFTWARE DOCUMENTATION IS NEEDED TO SERVE THE FOLLOWING PURPOSES:

1. PROVIDE THE PERFORMANCE REQUIREMENTS THAT DRIVE THE DESIGN AND DEVELOPMENT EFFORT.
2. OUTLINE THE TESTING NECESSARY TO DEMONSTRATE QUALITY AND COMPLIANCE WITH REQUIREMENTS.
3. PROVIDE WRITTEN COMMUNICATION ABOUT THE DESIGN AND ITS REQUIREMENTS SO THAT THE TASK IS BETTER UNDERSTOOD AND THUS A HIGHER QUALITY PRODUCT IS PRODUCED.
4. PROVIDE THE INFORMATION REQUIRED TO MANAGE THE SOFTWARE EFFORT EFFECTIVELY.
5. SUPPLY ALL INFORMATION REQUIRED TO MAINTAIN THE FINISHED PRODUCT.
6. PROVIDE THE DATA BASE REQUIRED TO PERFORM CHANGE CONTROL AND QUALITY ASSURANCE FUNCTIONS.

TO MEET THE REQUIRMENTS FOR GOOD DOCUMENTATION AND FOR CERTIFICATION SEVERAL DOCUMENTS AND ASSOCIATED DOCUMENT NUMBERS HAVE BEEN IDENTIFIED. THESE DOCUMENT NUMBERS ALONG WITH DOCUMENT TITLES, A VERY BRIEF DESCRIPTION, AND RESPONSIBILITY FOR PREPARATION ARE LISTED BELOW. ALL DOCUMENTS ARE REVISION CONTROLLED.

1. SYSTEM CONFIGURATION INDEX -- 000-XXXX-RN

   THE SYSTEM CID SHOWS ALL PRINCIPLE LRUS AND INTERFACES FOR THE SYSTEM. THE SYSTEM CID IS NOT A "FLAVOR CHART". IT MUST SHOW THE EXACT CONFIGURATION AND VERSION LEVEL FOR ALL LRUS.
   PREPARED BY: JOINT EFFORT OF MANAGEMENT, PROJECT LEADER, SYSTEM ENGINEERING, AND DESIGN TEAM.

2. SYSTEM SPECIFICATION -- 701-XXXX-RN

   THE SYSTEM SPEC IS A HIGH-LEVEL DESCRIPTION OF THE PRODUCT. IT SHOULD CONTAIN ALL OPERATIONAL, INTERFACE, AND PERFORMANCE REQUIREMENTS AS WELL AS FUNCTIONAL BLOCK DIAGRAMS. INPUTS SHOULD BE OBTAINED FROM THE MANAGEMENT PLAN AND ALL MAJOR COMPANY ORGANIZATIONS INVOLVED IN PRODUCT DEFINITION.
   PREPARED BY: JOINT EFFORT OF MANAGEMENT, PROJECT LEADER, SYSTEM ENGINEERING, AND DESIGN TEAM.

3. SOFTWARE REQUIREMENTS DOCUMENT -- 705-XXXX-RN

   THE REQUIREMENTS DOCUMENT SPECIFIES ALL FUNCTIONS PERFORMED BY SOFTWARE IN ORDER TO MEET THE SYSTEM SPEC. ALL

REQUIREMENTS MUST BE TESTABLE.
PREPARED BY: SOFTWARE PROJECT ENGINEER AND SOFTWARE DESIGN TEAM.

4. SOFTWARE TEST PLAN -- 707-XXXX-RN

THE TEST PLAN FOR ALL SOFTWARE CONTAINED IN AN LRU. THE TEST PLAN MUST INDICATE HOW TESTING WILL BE DONE TO SATISFY CERTIFICATION AND TO VERIFY THAT ALL REQUIREMENTS HAVE BEEN MET.
PREPARED BY: SOFTWARE PROJECT ENGINEER AND SOFTWARE DESIGN TEAM.

5. SOFTWARE EXCUTIVE SUMMARY -- 708-XXXX-RN

SOFTWARE EXECUTIVE SUMMARY
PREPARED BY: PROJECT LEADER.

6. LRU SOFTWARE DOCUMENT -- 715-XXXX-RN

DOCUMENT CONTAINING LRU LEVEL DESCRIPTION OF SOFTWARE. A SEPARATE DOCUMENT IS REQUIRED FOR EACH LRU.
PREPARED BY: SOFTWARE PROJECT ENGINEER AND SOFTWARE DESIGN TEAM.

7. LRU SOFTWARE STRUCTURE DIAGRAM -- 716-XXXX-RN
SOFTWARE STRUCTURE DIAGRAM FOR EACH LRU.
PREPARED BY: PROJECT ENGINEER.

8. PROGRAMMER'S REFERENCE LITERATURE -- 718-XXXX-RN
DOCUMENT CONTAINING A BIBLIOGRAPHY OF PROGRAMMER'S REFERENCE LITERATURE. IT IS SPECIFIC TO EACH TIME OF MICROPROCESSOR.
PREPARED BY: SOFTWARE CONTROL.

9. SOURCE LISTING DOCUMENT -- 719-XXXX-RN
SOURCE LISTING DOCUMENT PREPARED FROM THE VAX TEXT FILES CONTAINING THE SOURCE CODES.
PREPARED BY: SOFTWARE CONTROL.

10. LOAD FILE RELEASE -- 720-XXXX-RN
LOAD FILE RELEASE DOCUMENT USED TO RECORD ALL THE INFORMATION NECESSARY TO CREATE THE EXACT VERSION OF TARGET SOFTWARE SPECIFIED. A SEPARATE DOCUMENT IS REQUIRED FOR EACH PROCESSOR.
PREPARED BY: SOFTWARE CONTROL.

11. DESIGN DESCRIPTION -- 721-XXXX-RN
A COMPLETE AND DETAILED DESCRIPTION OF THE PRODUCT SOFTWARE. EACH PROCESSOR SYSTEM REQUIRES A SEPARATE DOCUMENT.
PREPARED BY: SOFTWARE PROJECT ENGINEER.

12. MACHINE CODE DOCUMENT -- 722-XXXX-RN
DOCUMENT CONTAINING A FORMATTED PRINTOUT OF THE LITERAL MACHINE CODE, IN ASCII REPRESENTATION, FOR THE PROGRAMMED MEMORY DEVICE. A SEPARATE DOCUMENT IS REQUIRED FOR EACH DEVICE.
PREPARED BY: SOFTWARE CONTROL.

13. COMPONENT SPECIFICATION -- 122-XXXX-RN
COMPONENT SPECIFICATION FOR PROGRAMMED MEMORY DEVICE (FIRMWARE).
PREPARED BY: COMPONENT ENGINEERING.

SOFTWARE REQUIREMENTS DOCUMENT

THE SOFTWARE REQUIREMENTS DOCUMENT IS A SYSTEM LEVEL DESCRIPTION OF WHAT THE SOFTWARE IS REQUIRED TO DO. IT SHOULD CONTAIN AS MUCH INFORMATION ABOUT OPERATIONAL AND COMPUTATIONAL REQUIREMENTS AS POSSIBLE AND SPELL OUT IN DETAIL THE HARDWARE/SOFTWARE INTERFACE REQUIREMENTS. A WELL PREPARED REQUIREMENTS DOCUMENT CAN IMPROVE THE QUALITY OF THE END PRODUCT AND SAVE DEVELOPMENT TIME AND MONEY AS WELL. IT SHOULD BE A 'LIVING' DOCUMENT THAT IS UPDATED TO REFLECT NEW REQUIREMENTS OR BETTER UNDERSTANDING OF PREVIOUS ONES. TO AVOID UNNECESSARY

IT SHOULD NOT BE UNDER REVISION CONTROL BE MAINTAINED BY THE PROJECT SYSTEM DESIGN GROUP

SOFTWARE SPECIFICATIONS ARE USED TO GUIDE THE DESIGN AND DEVELOPMENT OF PRODUCT SOFTWARE. A WELL PREPARED SET OF SPECIFICATIONS CAN SAVE A PROJECT CONSIDERABLE TIME AND COST. SOFTWARE SPECIFICATIONS ARE CONTAINED IN THE SOFTWARE REQUIREMENTS DOCUMENT. THE REQUIREMENTS DOCUMENT MUST BE A 'LIVING' DOCUMENT SO AS THE PRODUCT IS DEVELOPED, IT CAN BE CHANGED TO REFLECT NEW REQUIREMENTS AND IMPROVED INSIGHT INTO PREVIOUS REQUIREMENTS. IT SHOULD NOT BE RELEASED INTO REVISION CONTROL UNTIL LATE IN THE PROJECT. THIS IS DONE TO REMOVE THE BURDEN OF ECO OF THE DOCUMENT. EVERYONE SHOULD AGREE IN THE SPECIFICATIONS AT THE BEGINNING AND AT VARIOUS REVIEW INTERVALS.

SPECIFICATIONS SHOULD BE DEVELOPED TO THAT THEY ARE TESTABLE AND PROVIDE ENOUGH INFORMATION THE THE DESIGNERS CAN USE THEM TO DEVELOP DESIGN SPECIFICATIONS.

SPECIFICATIONS ARE REQURIEMENTS TURNED INTO SPECIFIC ACTION ITEMS

DESIGN AND DEVELOPMENT

SOFTWARE DESIGN AND DEVELOPMENT INVOLVES DEVELOPING A DESIGN PLAN BASED ON THE REQUIREMENTS DOCUMENT AND THEN CODING, TESTING AND INTEGRATING TO PRODUCE A COMPLETED AND TESTED PRODUCT.

SOFTWARE TESTING IN DEVELOPMENT

1. INTRODUCTION:

THE SUBJECT OF SOFTWARE TESTING WILL BE SEPARATED INTO TWO AREAS. THIS CHAPTER WILL ADDRESS TESTING DURING DESIGN AND DEVELOPMENT. A SUBSEQUENT CHAPTER WILL ADDRESS ACCEPTANCE TESTING BEFORE THE START OF PRODUCTION.

SOFTWARE TESTING IS PROBABLY THE MOST DIFFICULT TASK OF THE ENTIRE DESIGN AND DEVELOPMENT PROCESS. IT IS ALSO DIFFICULT TO PRESENT SUCH A COMPLEX ISSUE IN ANY KIND OF CONCISE WAY. ENTIRE TEXTBOOKS HAVE BEEN WRITTEN ON THE SUBJECT BUT THESE ARE GENERALLY TARGETTED TOWARD THE LARGE SYSTEM ENVIRONMENTS SUCH AS EDP OR OPERATING SYSTEMS AND HAVE NOT ADEQUATELY ADDRESSED THE NEEDS OF EMBEDDED SOFTWARE APPLICATIONS.

THE TASK OF SOFTWARE TESTING ACTUALLY SPANS THE ENTIRE DEVELOPMENT CYCLE OF A PRODUCT STARTING WITH THE REQUIREMENTS DOCUMENT AND ENDING WITH FINAL ACCEPTANCE TESTING. PROBABLY THE MOST INTENSE EFFORTS OCCUR DURING MODULE AND FUNCTION TESTING.

SOME TYPES OF SOFTWARE TESTING IS PERFORMED AS A DIRECT RESPONSE TO CERTIFICATION REQUIREMENTS AS SPECIFIED IN DO-178. OTHER TESTS ARE REQUIRED AS A MATTER OF GOOD SOFTWARE ENGINEERING PRACTICE. STILL OTHER TESTS MAY BE DONE AT THE OPERATIONAL LEVEL TO INSURE THAT THE EQUIPMENT PERFORMS THE INTENDED TASKS (I.E. DOES IT DO WHAT THE "CUSTOMER" WANTS?). FOR CERTIFICATION, DO-178 OUTLINES THE TESTING LEVELS REQUIRED FOR COMPLIANCE IN EACH CATAGORY OF CRITICALITY. TESTING FOR GOOD ENGINEERING PRACTICE WOULD INCLUDE EVALUATION OF SINGLE-ENTRY, SINGLE-EXIT MODULES TO SEE THAT ALL PATHS HAVE BEEN EXECUTED AND THAT ALL ANALOMOUS INPUTS ARE PROPERLY HANDLED.

2. TESTING METHODS:

METHODS FOR SOFTWARE TESTING VARY WIDELY ACCORDING TO THE TYPE OF PRODUCT, SUPPORT ENVIRONMENT, CERTIFICATION REQUIREMENTS, AND PERSONAL PREFERENCES OF THE PROJECT TEAM. THE FOLLOWING TYPES OF TESTING HAVE BEEN IDENTIFIED FOR OUR PRESENT ENVIRONMENT:

A) VAX HIGH-LEVEL-LANGUAGE TESTING.
B) VAX-SIMULATOR TESTING OF MACHINE CODE.
C) EMULATOR-ONLY
D) EMULATOR-TARGET
E) TARGET-ONLY

WE WILL NOW PRESENT DESCRIPTIONS OF THE BASIC TESTING METHODS AND DISCUSS SOME OF THE ISSUES ASSOCIATED WITH EACH. SEVERAL OR ALL OF THESE METHODS MIGHT BE USED ON ANY PARTICULAR PROJECT.

A) VAX HIGH-LEVEL-LANGUAGE TESTING.

WHEN THE PRODUCT SOFTWARE SOURCE CODE IS WRITTEN IN A HIGH LEVEL LANGUAGE AND MODULES BEING TESTED DO NOT REQUIRE SPECIFIC HARDWARE I/O INTERFACING, TESTING CAN BE DONE ON THE VAX. THIS IS DONE BY WRITING HLL CUSTOM TEST SOFTWARE THAT SERVES AS INPUT AND OUTPUT FOR THE MODULE. THE MODULE CAN BE TESTED TO THE DESIRED CERTIFICATION LEVEL AND THE TESTS CAN BE EASILY DOCUMENTED WITH VAX PRINTOUTS. THE BEST EXAMPLE OF AN APPLICATION FOR THIS TYPE OF TESTING IS FOR MATHMATICAL PROCESSING SUCH AS USED FOR NAVIGATION COMPUTATIONS. ADVANTAGES OF HLL SOFTWARE TESTING INCLUDE:

1) DETAILED MODULE TESTS CAN BE PERFORMED WITHOUT DEPENDENCE ON ACCURACY OR AVAILABILITY OF SUPPORT SOFTWARE, EMULATORS, AND TARGET HARDWARE.
2) INTEGRATION EFFORT IS REDUCED BECAUSE SOFTWARE HAS BEEN EXTENSIVELY TESTED AND BUG LEVEL GREATLY REDUCED.
3) A BETTER FEEL FOR SOFTWARE PERFORMANCE CAN BE OBTAINED BECAUSE CUSTOM DRIVERS AND DISPLAY CAN EXERCISE THE MODULE IN WAYS NOT AVAILABLE ON THE EMULATOR OR TARGET.

DISADVANTAGES ARE:

1) COST AND TIME ASSOCIATED WITH WRITING THE CUSTOM SOFTWARE

ORIVERS AND DISPLAYS NEEDED FOR EACH MODULE. IT IS POSSIBLE TO SPEND AS MUCH EFFORT WRITING THE TEST SOFTWARE AS WAS SPENT WRITING THE PRODUCT CODE.

2) THE TEST SOFTWARE MUST BE DOCUMENTED AND ARCHIVED AS PART OF THE SUPPORT ENVIRONMENT.
3) IT IS DIFFICULT TO ACCURATELY TEST MODULES THAT CONTAIN TARGET MACHINE I/O.
4) MODULES WRITTEN IN ASSEMBLY LANGUAGE CAN ONLY REASONABLY BE TESTED USING THE SIMULATOR.
5) IT IS DIFFICULT TO TEST THE TIMING INTERACTIONS OF CRITICAL TASKS THAT MUST BE EXECUTED UNDER EXECUTIVE CONTROL.

B) VAX-SIMULATOR TESTING OF MACHINE CODE.

MODULE TESTING ON THE VAX HOST CAN ALSO BE DONE USING THE MICROPROCESSOR HARDWARE SIMULATOR. WITH THIS TECHNIQUE, THE SOURCE MODULES TO BE TESTED ARE FIRST CONVERTED TO MACHINE CODE. THE USER THEN 'EXECUTES' THE MACHINE CODE BY RUNNING THE INTERACTIVE SIMULATOR. THE SIMULATOR PROVIDES DETAILED INFORMATION ON MACHINE ACTIVITY INCLUDING REGISTER CONTENTS AND TIMING. ADVANTAGES OF SIMULATOR TESTING OF SOFTWARE INCLUDE:

1) PROVIDES DETAILED TESTING OF MACHINE CODE ON VAX HOST. THIS REDUCES DEMAND ON EMULATORS AND TARGET HARDWARE SUPPORT.
2) PROVIDES EXCELLENT DETAILED, DOCUMENTED TEST DATA FOR MACHINE ACTIVITY.

DISADVANTAGES ARE:

1) SOFTWARE PEOPLE MAY NOT BE FAMILIAR WITH DETAILED WORKINGS OF TARGET PROCESSOR AND THEREFORE NOT ABLE TO INTERPRET RESULTS. ALSO, NUMERICAL RESULTS ARE GIVEN IN HEX SO THAT TIRESOME DECIMAL INTERPRETATION OF CALCULATIONS MUST BE DONE EACH TIME.
2) SIMULATOR TESTS RUNS ARE SLOW WHEN COMPARED TO EMULATOR TEST RUNS AND THE USER STILL MUST GO THROUGH THE SAME SLOW CYCLE OF EDIT COMPILE, ASSEMBLE, LOAD, AND DOWNLOAD FOR EACH SOFTWARE CHANGE AND SUMULATOR TEST RUN.

C) EMULATOR-ONLY SOFTWARE TESTING.

SOFTWARE CAN BE DOWNLOADED TO A HARDWARE EMULATOR FOR TESTING. THIS CAN BE A DESIRABLE ALTERNATIVE TO HOST TESTING IN MANY CASES. ADVANTAGES OF TESTING SOFTWARE WITH AN EMULATOR INCLUDE:

1) ELIMINATION OF NEED FOR WRITING HLL CUSTOM TEST SOFTWARE ON VAX HOST.
2) CODE IS EXECUTED IN ALMOST REAL TIME SO EACH EMULATION SESSION IS FASTER THAN WITH THE SIMULATOR ON VAX.
3) TESTS ARE RUN CLOSER TO THE ACTUAL TARGET MICROPROCESSOR ENVIRONMENT SO THAT GENERALLY THE TESTS HAVE MORE VALIDITY.
4) IF EMULATOR MICROPROCESSOR CLOCK SPEED AND DEBUG OVERHEAD ARE PROPERLY ACCOUNTED FOR, TIMING DATA CAN BE OBTAINED FROM ACTUAL MACHINE EXECUTION.
5) EMULATOR SESSIONS PROVIDE A CROSSCHECK FUNCTION FOR CORRECTNESS OF CODE PRODUCED BY COMPILERS, ASSEMBLERS, LOADERS, AND RUN-TIME LIBRARIES.
6) ADVANTAGE OVER TARGET-ONLY TESTING BECAUSE THERE IS MUCH MORE TESTING CAPABILITY AND YOU DON'T HAVE TO DEBUG NEW TARGET HARDWARE BEFORE SOFTWARE TESTING.

DISADVANTAGES OF EMULATOR USE FOR SOFTWARE TESTING INCLUDE:

1) TESTING MUST WAIT UNTIL MOST OF THE PRODUCT CODE CAN BE DOWNLOADED TO THE EMULATOR. THIS DELAYS TESTING AND PREVENTS THE NECESSARY EARLY FEEDBACK ON TIMING AND I/O PROBLEMS.
2) DESIGN AND DOCUMENTATION OF THE TESTING IS MORE DIFFICULT THAN HLL OR SIMULATOR TESTING ON VAX HOST. TESTS HAVE DO BE DONE USING EMULATOR OPERATION SYSTEM AND SOFTWARE AND DOCUMENTATION MAY NOT BE AUTOMATIC.
3) SLOW THROUGHPUT DO TO THE NEED TO GO THROUGH A CYCLE OF EDIT, COMPILE, ASSEMBLE, LOAD, AND DOWNLOAD FOR EACH SOFTWARE CHANGE AND TEST.
4) CAN'T DO TRUE I/O TESTING.
5) CAN'T DO ASYNCHRONOUS INTERRUPT TESTING.
6) CUSTOM SOFTWARE REQUIRED TO DO TESTING ON EMULATOR MUST NOW BE WRITTEN FOR TARGET MACHINE AND EMBEDDED IN MACHINE CODE. THIS INCREASES CODE WRITING OVERHEAD AND CAUSES PROBLEMS WHEN TEST CODE IS REMOVED FROM PRODUCT SOURCE MODULES.
7) EMULATORS USED FOR SOFTWARE TESTING ARE UNAVAILABLE FOR THEIR PRIMARY USE WHICH IS HARDWARE/SOFTWARE INTEGRATION.

8) WHEN TEST PROBLEMS OCCUR, IT IS DIFFICULT TO TRACE THE TRUE CAUSE. THE PROBLEM MAY BE CAUSED BY PRODUCT CODE BUGS, EMBEDDED TEST CODE BUGS, COMPILER AND RUNTIME ERRORS, OR EMULATOR HARDWARE/SOFTWARE ERRORS. OF COURSE, THE SOFTWARE DESIGNER IS LOOKING FOR BUGS IN THE PRODUCT CODE BUT IS OFTEN FACED WITH FINDING AND ISOLATING PROBLEMS IN THE OTHER AREAS AS WELL.

D) EMULATOR-TARGET SOFTWARE TESTING.

SOFTWARE TESTING WITH BOTH THE EMULATOR AND TARGET MICROPROCESSOR IS MORE PROPERLY REFERRED TO AS SOFTWARE/HARDWARE INTEGRATION. INTEGRATION TESTING OFTEN IS DONE IS SPURTS OF SOFTWARE TESTS, THEN HARDWARE TESTS. HOPEFULLY, MOST OF THE DETAILED TESTS OF EACH HAVE BEEN COMPLETED PRIOR TO INTEGRATION. THIS PRELIMINARY TESTING IS VITAL IF THE PROJECT IS TO AVOID COSTLY DELAYS CAUSED BY DETAILED TESTING PROBLEMS IN THE INTEGRATION PHASE. THE PRODUCT HARDWARE CAN BE TESTED BY WRITING SIMPLE SOFTWARE ROUTINES ON THE VAX HOST AND DOWNLOADING THEM TO THE EMULATOR OR TARGET PROM TO AID IN HARDWARE DEBUG. WITH THIS TECHNIQUE, MOST HARDWARE BUGS CAN BE CORRECTED BEFORE INTEGRATION STARTS AND THUS AVOID MANY 'IS IT HARDWARE OR SOFTWARE' QUESTIONS THAT ARIZE.

E) TARGET-ONLY SOFTWARE TESTING.

DETAILED TESTING OF SOFTWARE ON ONLY THE TARGET MACHINE SHOULD BE GENERALLY AVOIDED EXCEPT IN SOME SPECIAL CASES. THIS TYPE OF TESTING COULD BE DONE EFFECTIVELY FOR SINGLE CHIP PROCESSORS USED FOR I/O INTENSIVE APPLICATIONS. IN SOME PRODUCTS, THE TESTING REQUIREMENTS MAY ONLY CALL FOR OPERATIONAL TESTS AND FINAL ACCEPTANCE TESTING AND IT MAKES GOOD SINCE TO DO THESE WITH ONLY THE TARGET PROCESSOR. OF COURSE, OPERATIONAL TESTS SHOULD ALWAYS BE DONE ON THE TARGET WITHOUT ANY ASSISTANCE FROM HOST OR EMULATOR.

****
**888 18 JUNE 1983 STEVE RUSSELL
I NEED TO WRITE UP A DISCUSSION OF EACH OF THE TOPICS LISTED BELOW BUT I DONT KNOW HOW TO ORGANIZE THEM.. PHASES OR CATAGORIES DONT SEEM TO BE RIGHT. I AM GOING TO SLEEP ON IT.
****

3. TEST PHASES:

SEVERAL PHASES CAN BE IDENTIFIED IN THE TESTING SCENARIO. THESE WILL BE IDENTIFIED AND DESCRIBED.

A) REQUIREMENTS PHASE
B) TEST PLAN PHASE
C) MODULE DEFINITION PHASE
D) TESTING LEVELS
E) MODULE TESTS
F) FUNCTIONAL LEVEL TESTS
G) SOFTWARE/HARDWARE INTEGRATION
H) OPERATIONAL PERFORMANCE TESTING
I)

**888

SOFTWARE TESTING FOR EMBEDDED APPLICATIONS CAN BE CATAGORIZED AS FOLLOWS:

1. REQUIREMENTS DOCUMENT - TESTING MUST BE KEPT IN MIND WHEN WRITING THE SOFTWARE REQUIREMENTS DOCUMENT SO THAT TESTABLE REQUIREMENTS ARE GENERATED. EVERY REQUIREMENT MUST BE TESTABLE TO BE OF VALUE.

2. TEST PLAN - THE TEST PLAN SHOULD BE WRITTEN SO THAT A TEST IS PERFORMED TO VERIFY THAT EVERY REQUIREMENT HAS BEEN MET. THE FIRST DRAFT OF THE PLAN USUALLY IS SKETCHY BECAUSE THE DESIGN ARCHITECTURE AND MODULE DEFINITIONS ARE INCOMPLETE. AS THE DESIGN NEAR COMPLETION, THE TEST PLAN CAN BE REVISED TO COVER ALL THE REQUIREMENTS. THE TEST PLAN MUST COVER NOT ONLY WHAT TESTS ARE TO BE RUN BUT ALSO WHAT THE EXPECTED RESULTS WILL BE AND HOW TO DETERMINE A PASS OR FAIL. THE TESTS FOR SOME REQUIREMENTS CAN BE DONE AT THE MODULE LEVEL WHILE OTHERS MUST BE DONE AT THE FUNCTIONAL LEVEL OR EVEN AT SYSTEM VALIDATION. THE TEST PLAN WILL INCLUDE TESTS OTHER THAN JUST THOSE REQUIRED TO SHOW COMPLIANCE WITH THE SPECIFICATION. GENERALLY, THESE WILL BE MODULE-LEVEL TESTS DONE TO SHOW THAT THE MODULES HAVE WELL BEHAVED AND

PREDICTABLE OUTPUTS FOR ALL POSSIBLE RANGES OF INPUTS.

3. MODULE DEFINITION - THE REQUIREMENTS FOR TESTING WILL INFLUENCE THE SOFTWARE ARCHITECTURE AND HOW MODULES ARE DEFINED. A DESIGN WHICH MAKES GOOD FUNCTIONAL SENSE MIGHT NOT BE TESTABLE TO THE REQUIRED LEVEL. THE ARCHITECTURE AND SUBSEQUENT BREAKDOWN INTO MODULES WILL HAVE TO BE A COMPROMISE BETWEEN FUNCTION AND TESTING.

4. TESTING LEVELS - TESTING WILL OCCUR AT THREE LEVELS.
   A) MODULES LEVEL TESTS.
   B) FUNCTIONAL LEVEL TESTS (GROUPS OF MODULES).
   C) VALIDATION TESTS (OPERATIONAL TESTS OF ENTIRE PRODUCT)

5. MODULE TESTING - MODULES CAN BE TESTED EITHER ON THE HOST COMPUTER OR THE EMULATOR. IF TESTING IS DONE ON THE HOST COMPUTER, THE MODULES CAN BE PROCESSED WITH THE SIMULATOR OR CUSTOM INTERFACE SOFTWARE MAY BE WRITTEN TO DO I/O TESTING. MODULE TESTING ON THE EMULATOR IS NOT AS SELF DOCUMENTING BUT USUALLY IS FASTER AND IS OFTEN THE ONLY PRACTICAL WAY TO TEST CRITICAL TIMING FEATURES. MODULE TESTING SHOULD MAINLY CONCENTRATE ON MAKING SURE ALL SOFTWARE PATHS HAVE BEEN EXECUTED (NOT NECESSARILY EXHAUSTIVE PATH TESTING) AND THAT ALL RANGES OF PROPER AND IMPROPER INPUTS RESULT IN PREDICTABLE OUTPUTS.

6. FUNCTIONAL LEVEL TESTING - ONCE THE COLLECTION OF MODULES USED TO PERFORM A SPECIFIC FUNCTION HAS BEEN WRITTEN AND TESTED, THEY MAY BE LINKED AND THE FUNCTION CAN BE TESTED. IN MANY CASES, THIS IS THE FIRST LEVEL OF TESTING THAT CAN BE RELATED DIRECTLY TO SOFTWARE REQUIREMENTS. FUNCTIONAL TESTING CAN ALSO BE DONE ON THE HOST OR ON THE EMULATOR. IN MOST CASES EMULATOR TESTING WILL BE USED BECAUSE THE FUNCTIONS TEND TO BE RELATED TO TASKS THAT HAVE A COMPLEX HARDWARE INTERACTION.

7. HARDWARE SOFTWARE INTEGRATION.

7. OPERATIONAL LEVEL TESTING - WHEN ENOUGH OF THE SOFTWARE HAS BEEN DESIGNED

AFTER THE SOFTWARE HAS ESSENTIALLY

SYSTEM INTEGRATION

INTEGRATION IS THE PROCESS OF DEBUGGING AND TESTING THE COMBINATION OF HARDWARE AND SOFTWARE. A VARIETY OF TECHNIQUES HAVE BEEN DEVELOPED TO DO THIS AND NEW ONES ARE CONSTANTLY BEING TRIED. THE CHOICE OF A PARTICULAR TECHNIQUE WILL DEPEND ON THE SIZE OF THE SOFTWARE AND HOW IT IS USED.

PAST STUDIES HAVE SHOWN THAT INTEGRATION IS PROBABLY THE SINGLE MOST COSTLY PART OF A SOFTWARE PROJECT BECAUSE IT IS SO COMPLICATED AND TIME CONSUMING. FOR THIS REASON, ANYTHING THAT CAN BE DONE TO REDUCE INTEGRATION TIME WILL BE COST EFFECTIVE. THIS INCLUDES CHOOSING THE BEST TECHNIQUE FOR THE PROJECT, HAVING THE VERY BEST TOOLS AVAILABLE, AND DOING ALL THE PRELIMINARY PREPARATIONS NEEDED TO SUPPORT INTEGRATION.

BEFORE BEGINNING INTEGRATION, THE FOLLOWING PRELIMINARY WORK SHOULD BE DONE:

1. PRODUCT SOFTWARE SHOULD BE TESTED AND VALIDATED AT THE MODULE LEVEL AND AT THE MODULE INTEGRATION LEVEL. THIS WILL ELIMINATE OBVIOUS PROBLEMS FROM SHOWING UP DURING INTEGRATION.
2. CUSTOM TEST SOFTWARE THAT WILL BE USED FOR THE PRODUCT CODE SHOULD ALREADY BE PLANNED, DEVELOPED, AND TESTED SO THIS ACTIVITY WILL NOT SLOW DOWN THE INTEGRATION EFFORT. THIS INCLUDES CUSTOM TEST CODE LINKED WITH THE PRODUCT CODE AS WELL AS CUSTOM TEST CODE THAT MAY BE HOSTED ON A COMPUTER DRIVEN TEST BED.
3. DESIGNERS ACTUALLY USING THE INTEGRATION TEST FACILITY SHOULD BE TRAINED ON ITS USE PRIOR TO INTEGRATION.
4. THE TARGET HARDWARE SHOULD BE FULLY DEVELOPED AND TESTED TO ELIMINATE OBVIOUS PROBLEMS. THIS CAN BE DONE BY WRITING SIMPLE TEST SOFTWARE ROUTINES, TO BE DOWNLOADED TO EMULATOR OR EPROM, THAT WILL EXERCISE ALL HARDWARE FUNCTIONS.
5. THE PROJECT SCHEDULE SHOULD BE PLANNED SO THAT MEMBERS OF THE DEVELOPMENT TEAM NOT DIRECTLY INVOLVED WITH THE DAY-TO-DAY INTEGRATION ACTIVITIES WILL HAVE ALTERNATE ACTIVITIES TO FILL IN THE SLACK TIMES WHEN THEY ARE NOT DIRECTLY SUPPORTING THE INTEGRATION EFFORT.
6. THE INTEGRATION TEST FACILITY SHOULD BE PLANNED AND MADE AVAILABLE PRIOR TO WHEN INTEGRATION IS SCHEDULED TO BEGIN.
7. BOTH THE HARDWARE AND SOFTWARE FOR ANY CUSTOM TEST EQUIPMENT USED TO DRIVE EITHER ANALOG OR DIGITAL PORTS SHOULD BE DEVELOPED AND READY TO BE USED AT INTEGRATION TIME.

THERE ARE THREE BASIC ENVIRONMENTS IN WHICH INTEGRATION IS COMMONLY DONE ALTHOUGH THERE ARE TIMES WHEN A MIXTURE OF THESE MAY BE USED. THESE ENVIRONMENTS ARE:

1. PRODUCT HARDWARE IN OPERATIONAL CONFIGURATION.
2. PRODUCT HARDWARE WITH EMULATOR.
3. PRODUCT HARDWARE WITH INTEGRATION TEST BED.

IN ALL OF THESE ENVIRONMENTS, CUSTOM TEST EQUIPMENT MAY BE NEEDED TO SIMULATE ANALOG AND DIGITAL I/O WITH THE PRODUCT UNDER TEST. THIS MAY BE A TEST SET USED BY THE FACTORY OR IT MAY BE PROJECT-DEVELOPED OR IT MAY BE PURCHASED OUTSIDE. FOR LARGE SYSTEMS, THE CTE MAY INVOLVE SIGNAL GENERATORS AND DIGITAL I/O THAT ARE DRIVEN BY CUSTOM SOFTWARE ON A MINICOMPUTER.

THE FIRST APPROACH TO INTEGRATION USES ONLY THE PRODUCT HARDWARE AND SOFTWARE TO DO THE BASIC TESTS. INPUT/OUTPUT IS PERFORMED WITH THE PRODUCT CONTROL/DISPLAY FUNCTIONS. USUALLY, A LOGIC ANALYZER IS USED TO SET SIMPLE BREAKPOINTS AND STUDY BUS ACTIVITY. SOMETIMES MEANINGFUL TEST CODE CAN BE EMBEDDED IN THE PRODUCT CODE BUT THIS IS USUALLY NOT DONE BECAUSE OF LIMITED I/O CAPABILITY. MACHINE CODE IS DOWNLOADED FROM THE HOST DIRECTLY TO TARGET MEMORY (USUALLY EPROM BUT SOMETIMES NONVOLATILE RAM). THIS ENVIRONMENT IS SIMPLE TO IMPLEMENT BUT IT IS VERY RESTRICTED AND

PROVIDES THE LEAST CAPABILITY TO DO SOFTWARE TESTING. IT SHOULD BE USED ONLY IN APPLICATIONS THAT ARE LARGELY CONTROL ORIENTED, INVOLVE NO MORE THAN 1-4K OF CODE, AND HAVE CONTROL/DISPLAY FUNCTIONS THAT EXERCISE ALL OF THE CODE.

THE SECOND APPROACH IS CURRENTLY THE MOST FREQUENTLY USED FOR AVERAGE-COMPLEXITY PROJECTS. HERE, THE CODE IS DOWNLOADED THROUGH THE EMULATOR AND TESTS ARE PERFORMED USING THE CAPABILITIES OF THE EMULATOR. EMULATOR TESTING CAPABILITIES INCLUDE:

1. SETTING COMPLICATED BREAKPOINT SCHEMES AND CAPTURING AND DISPLAYING BUS ACTIVITY, PROGRAM COUNTER, STACK, ETC.
2. HALTING THE BUS AND EXAMINING ALL ADDRESSES, MACHINE REGISTERS, ETC.
3. HALTING EXECUTION AND DISASSEMBLING PREVIOUSLY EXECUTED CODE.
4. SYMBOLIC REFERENCING (SYMBOLIC DEBUG).
5. RUNNING OF PRODUCT CODE USING ONLY THE EMULATION PROCESSOR AND RAM IN THE EMULATOR. LIMITED SOFTWARE TESTING CAN BE DONE WITHOUT THE TARGET PROCESSOR HARDWARE.
6. DISPLAY/KEYBOARD INTERFACE TO CUSTOM TEST SOFTWARE THAT IS EMBEDDED IN THE PRODUCT SOFTWARE.

INTEGRATION USING AN EMULATOR PROVIDES MUCH MORE INFORMATION THAN THE FIRST TECHNIQUE BUT ACCESS IS STILL LIMITED FOR MANY SOFTWARE PERFORMANCE DETAILS. THE MOST UNDESIRABLE LIMITATIONS ARE:

1. THE BUS MUST BE HALTED TO EXAMINE DATA. MANY TIMES IT IS DESIRABLE TO EXAMINE DATA WHILE THE PROCESSOR CONTINUES TO RUN. IT IS DIFFICULT TO GET A FEEL FOR EITHER INTERACTIVE OR DYNAMIC PERFORMANCE.
2. CUSTOM TEST SOFTWARE MUST BE ACCESSED THROUGH SERVICE CALLS THAT ARE PART OF THE EMULATOR OPERATING SYSTEM. THESE ARE COMPLICATED TO USE AND HAVE LIMITED CAPABILITY.
3. CUSTOM TEST SOFTWARE IS EXECUTED IN THE RUN STREAM ALONG WITH THE PRODUCT CODE. THIS SLOWS EXECUTION AND CAN INTRODUCE ADDITIONAL BUGS THAT HAVE NOTHING TO DO WITH THE PRODUCT. ALSO, WHEN INTEGRATION IS COMPLETED, THE TEST CODE THAT HAS BEEN LINKED WITH THE PRODUCT CODE MUST NOW BE REMOVED. THIS RESULTS IN A NEW ABSOLUTE OBJECT MODULE THAT HAS NEVER BEEN TESTED.
4. I/O IS LIMITED TO KEYBOARD AND CRT DISPLAY. THERE ARE TIMES WHEN IT IS DESIRABLE TO "INSTRUMENT" THE SOFTWARE AND DYNAMICALLY RECORD PORT ACTIVITY, TRACKING LOOP ERRORS, ETC WITH STRIP CHART RECORDERS OR MAG TAPE RECORDERS. THIS CANNOT BE DONE WITH PRESENT EMULATORS.
5. CUSTOM TEST SOFTWARE MUST BE WRITTEN IN THE LANGUAGES AVAILABLE FOR THE TARGET MICROPROCESSOR. IF NO HLL SUPPORT IS AVAILABLE THEN ASSEMBLY LANGUAGE MUST BE USED.
6. "QUICK TEST" MACHINE CODE PATCHING WITHOUT GOING THROUGH A COMPLETE CYCLE OF SOURCE EDIT, COMPILE, ASSEMBLE, LINK, AND DOWNLOAD.

THE THIRD APPROACH INVOLVES THE DEVELOPMENT OF A FULL CUSTOM SOFTWARE TEST BED. THIS IS ESSENTIALLY A SEPARATE COMPUTER SYSTEM WHICH:

1. INTERFACES DIRECTLY TO THE TARGET PROCESSOR BUS.
2. IS THE BUS MASTER AND HAS DIRECT MEMORY ACCESS.
3. PERFORMS THE SAME FUNCTIONS NORMALLY PROVIDED BY AN EMULATOR THROUGH A DISPLAY/KEYBOARD INTERFACE.
4. HOSTS A GOOD OPERATING SYSTEM AND HAS ALL THE SUPPORT TOOLS NECESSARY TO WRITE CUSTOM TEST PROGRAMS THAT MAY BE EXECUTED INDEPENDENT OF THE TARGET.
5. INTERFACES OTHER TEST EQUIPMENT SUCH AS MAG TAPE AND STRIP CHART RECORDERS, PRINTERS, SOFTWARE DRIVEN CTE, AND OTHER PRODUCT EQUIPMENTS.

THE CUSTOM TEST BED IS THE BEST ENVIRONMENT FOR LARGE PROJECTS BECAUSE IT ALLOWS MUCH BETTER SOFTWARE TESTING AND RECORDING CAPABILITY. ITS MAJOR DRAWBACK IS THAT IT IS A SEPARATE DEVELOPMENT EFFORT THAT MUST

BE PLANNED, MANAGED, AND MANNED. DESIRABLE FEATURES OF THE TEST BED ENVIRONMENT INCLUDE:

1. THE ABILITY TO RUN COMPLEX TEST PROGRAMS WITHOUT IMPACTING THE RUN TIME OF THE TARGET PROCESSOR.
2. EXAMINATION OF MEMORY ADDRESSES WITHOUT HALTING THE TARGET PROCESSOR BUS.
3. DYNAMIC CHANGING OF DATA CONSTANTS WITHOUT REVISING SOFTWARE OR HALTING THE BUS.
4. DYNAMIC READING OF ANY MEMORY ADDRESSES AND RECORDING OF THE DATA WITHOUT HALTING THE BUS.
5. "QUICK TEST" MACHINE CODE PATCHING WITHOUT GOING THROUGH A COMPLETE CYCLE OF SOURCE EDIT, COMPILE, ASSEMBLE, LINK, AND DOWNLOAD.
6. DISPLAY OF VARIABLES IN DECIMAL FORMAT.

UNDESIRABLE CHARACTERISTICS INCLUDE:

1. COST OF DEVELOPING AND MAINTAINING THE TEST BED.
2. TURNAROUND TIME ASSOCIATED WITH MAKING REVISIONS TO TEST BED HARDWARE OR SOFTWARE.
3. LACK OF AVAILABILITY FROM OUTSIDE VENDORS. THE SYSTEM CAN BE DEVELOPED IN-HOUSE FROM STANDARD COMPONENTS BUT THE DMA INTERFACE AND OPERATING SYSTEM SOFTWARE MUST BE CUSTOM DESIGNED AND DEVELOPED.
4. DESIGN REQUIRES OPERATION FROM A BATTERY SOURCE SO THE TEST BED CAN BE USED FOR FLIGHT TESTS.

TO CONCLUDE, WE CAN MAKE SOME GENERAL COMMENTS CONCERNING SYSTEM HARDWARE AND SOFTWARE INTEGRATION. FIRST, THERE ARE SEVERAL SOFTWARE TASKS TO BE PERFORMED AND HENCE SEVERAL TYPES OF SOFTWARE. THE MOST COMMON ARE:

1. PRODUCT SOFTWARE.
2. EMULATOR OPERATING SYSTEM SOFTWARE.
3. TEST SOFTWARE EMBEDDED IN THE PRODUCT SOFTWARE.
4. TEST SOFTWARE HOSTED ON THE TEST BED PROCESSOR.
5. TEST BED PROCESSOR OPERATING SYSTEM SOFTWARE.
6. SOFTWARE USED TO DRIVE ANALOG AND DIGITAL SIGNAL SIMULATORS.
7. SOFTWARE USED TO PROCESS RECORDED DATA.

WHAT MUST BE KEPT IN MIND IS THAT EACH TYPE OF SOFTWARE MUST BE DESIGNED, DEVELOPED AND MAINTAINED. ALL HAVE THE POTENTIAL OF HAVING BUGS THAT IMPACT THE INTEGRATION PROCESS. SOME MAY BE DEVELOPED IN-HOUSE BUT OTHERS MAY BE THIRD-PARTY AND HENCE NOT REVISABLE WITHOUT CONSIDERABLE DELAY.

INITIAL RELEASE

SOFTWARE MAINTANENCE

THE TERM SOFTWARE MAINTANENCE WILL BE USED TO REFER TO ALL OF THE PROCESSES REQUIRED TO CHANGE ESTABLISHED SOFTWARE. THESE PROCESSES ARE:

1. IDENTIFICATION OF THE NEED FOR A CHANGE TO EXISTING SOFTWARE PRODUCT.
2. APPROVAL FOR SOFTWARE ENGINEERING EFFORT TO MODIFY EXISTING CODE.
3. RESTORATION OF SUPPORT ENVIRONMENT AND SOURCE CODE REQUIRED TO MAKE MODIFICATIONS.
4. DESIGN AND DEVELOPMENT EFFORT TO CHANGE SOFTWARE TO MEET NEW REQUIREMENT.
5. CONDUCT NEW ACCEPTANCE TESTS.
6. OBTAIN MANAGEMENT APPROVAL OF TESTS AND AUTHORIZATION TO RELEASE NEW VERSION.
7. ARCHIVING OF REVISED SOURCE CODE.
8. ARCHIVING OF REVISED SUPPORT ENVIRONMENT.
9. REVISION OF DOCUMENTATION.
10. RELEASE OF NEW VERSION TO THE FACTORY.

SOFTWARE ACCEPTANCE TESTING

INFORMATION ON SOFTWARE ACCEPTANCE TESTING CAN BE FOUND IN THE FOLLOWING REFERENCES:

BEIZER, BORIS "SOFTWARE TESTING TECHNIQUES", NEW YORK: VAN NOSTRAND REINHOLD COMPANY, 1983.

DUNN, ROBERT AND RICHARD ULLMAN, "QUALITY ASSURANCE FOR COMPUTER SOFTWARE", NEW YORK: MCGRAW-HILL BOOK COMPANY, 1982.

JENSON, RANDALL W. AND CHARLES C. TONIES, "SOFTWARE ENGINEERING", ENGLEWOOD CLIFFS, N.J.: PRENTICE-HALL, INC., 1979.

MILLER, EDWARD AND WILLIAM E. HOWDEN, "TUTORIAL: SOFTWARE TESTING AND VALIDATION TECHNIQUES", NEW YORK: IEEE COMPUTER SOCIETY PRESS, 1981.

MYERS, GLENFORD J., "SOFTWARE RELIABILITY", NEW YORK: JOHN WILEY AND SONS, INC., 1976.

RTCA (RADIO TECHNICAL COMMISION FOR AERONAUTICS), "SOFTWARE CONSIDERATIONS IN AIRBORNE SYSTEMS AND EQUIPMENT CERTIFICATION", DOCUMENT NO. RTCA/DO-178, NOVEMBER, 1981.

THE FORMAL TESTS CONDUCTED ON A PRODUCT AT THE END OF DESIGN AND DEVELOPMENT BUT PRIOR TO RELEASE INTO THE FACTORY ARE COMMONLY REFERED TO AS ACCEPTANCE (OR QUALIFICATION) TESTS. FOR SOFTWARE, THESE TESTS MUST DEMONSTRATE THAT ALL SPECIFICATIONS CALLED OUT IN THE REQUIREMENTS DOCUMENT HAVE BEEN SATISFIED.

AN INDEPENDENT QA TEAM IS OFTEN USED TO CONDUCT ACCEPTANCE TESTS USING A TEST PLAN WRITTEN BY THE DESIGN GROUP. THIS IS DONE TO INSURE THAT THE TESTS ARE CONDUCTED IMPARTIALLY. WHEN THE SOFTWARE IS BEING DEVELOPED UNDER CONTRACT, THE CUSTOMER WILL ALSO PROVIDE SOME PERSONNEL FOR THE ACCEPTANCE TEST TEAM. IN SOME CASES, A TEAM CAN BE FORMED FROM THE DESIGN GROUP TO DO THE TESTS AS LONG AS THEY ARE AUTONOMOUS UNTIL THE TEST RESULTS ARE COMPLETED. MANAGEMENT MUST REVIEW AND APPROVE ACCEPTANCE TEST RESULTS PRIOR TO FINAL SOFTWARE RELEASE.

THE ACCEPTANCE TEST PLAN IS PART OF THE TEST PLAN DOCUMENT THAT ALSO INCLUDES THE PLANS AND RESULTS OF DETAILED TESTING DURING DESIGN AND DEVELOPMENT. ULTIMATELY, THE TEST PLAN MUST ADDRESS THE DIFFICULT PROBLEM OF ESTABLISHING THE CRITERION FOR ACCEPTING OR REJECTING THE SOFTWARE. THE PASS/FAIL CRITERIA SHOULD BE STRUCTURED IN SUCH A WAY AS TO WEIGHT THE IMPORTANCE OF SPECIFICATIONS. THIS WILL AVOID REJECTION OVER TRIVIAL VARIANCES IN RESULTS. THE TEST PLAN SHOULD ALSO ALLOW FOR MINOR CHANGES USING APPROVED PROCEDURES AND THEN REGRESSION TESTING IF NECESSARY. MAJOR CHANGES REQUIRE REPEATING SOME OF THE DESIGN/DEVELOP/TEST CYCLE AND REDOING ACCEPTANCE TESTS FROM THE BEGINNING. THE TEST TEAM, DESIGN TEAM, MANAGEMENT, AND CUSTOMER MUST REVIEW RECOMMENDED CHANGES AND REACH AN AGREEMENT ON THEIR CLASSIFICATION AS MAJOR OR MINOR.

SOFTWARE CONFIGURATION MANAGEMENT

SOFTWARE QUALITY ASSURANCE

THE SOFTWARE QUALITY ASSURANCE ORGANIZATION MUST BE INDEPENDENT OF EITHER MANUFACTURING OR ENGINEERING.

SOFTWARE QUALITY DOES NOT DEGRADE WITH USE, IT IMPROVES AS BUGS FOUND AND CORRECTED.

IN HARDWARE, A LOT OF QA IS DEVOTED TO MAKING SURE THAT THE DESIGN IS BEING MAINTAINED. IN SOFTWARE THIS IS NOT NEEDED. THE CHIPS CAN BE PERFECTLY CLONED. IN SOFTWARE IT IS THE PROBLEM OF SEEING THAT THE DESIGN HAS A MINIMUM OF BUGS, KNOWN CONFIGURATION, AND REVISION CONTROL.

NO "STANDARD" COMPONENTS.
FUNCTIONS OF QA:

1. SEE THAT DESIGN AND CODING STANDARDS ARE BEING FOLLOWED.
2. EVALUATE TEST PLAN TO SEE IF IT MEETS CRITICALITY CATEGORY.
3. MONITOR ACCEPTANCE TESTS.
3. VERIFY THAT APPROVED CONFIGURATION MANAGEMENT AND RELEASE PROCEDURES ARE BEING FOLLOWED.

SOFTWARE DESIGN AND CODING STANDARDS

AT THE PRESENT TIME, THERE ARE NO DESIGN AND CODING STANDARDS BEING ENFORCED BY AN INDEPENDENT QUALITY AUDIT GROUP. EACH PROJECT TEAM HAS DETERMINED WHAT SIMPLE STANDARDS WILL BE MEET AND INFORMALLY INFORCED. CODING GUIDELINES WILL ALSO BE INCLUDED WITH CODING STANDARDS. METHODS TO WORK AROUND STANDARDS WE REAL PROBLEMS ARE ENCOUNTERED ALSO NEED TO BE DEVELOPED. WE DO NOT PLAN TO HAVE A SUPPORT SOFTWARE CAPABILITY FOR STANDARDS CHECKING UNTIL OUR STANDARDS HAVE BEEN PROVEN TO BE EFFECTIVE.

DESIGN AND CODING STANDARDS FOR THE FUTURE WILL PROBABLY INCLUDE THE FOLLOWING:

1. NAMING CONVENTIONS.
2. MODULE DEFINITIONS AS TO SIZE AND I/O.
3. USE OF STRUCTURE.
4. LINK AND LOAD PROCEDURES.
5. USE OF RUN TIME LIBRARIES.
6. INTEGRATION OF HLL AND ASSEMBLY-LEVEL MODULES.

SOME DOS AND DON'TS:

1. DON'T TRY TO BE CLEAVER OR USE CONVOLUTED LOGIC TO SAVE A FEW LINES OF SOURCE CODE OR BYTES OF MACHINE CODE. THE DIFFICULTY ENCOUNTERED IN MAINTANENCE WILL MORE THAN OFFSET ANY MEMORY SAVINGS.
2. USE STANDARD CONTROL CONSTRUCTS WHEN AT ALL POSSIBLE, THESE ARE:
   A) SEQUENCE
   B) IF-THEN-ELSE
   C) CASE
   D) DO-WHILE
   E) DO-UNTIL
3. USE UNCONDITIONAL GO TO IN A DISCIPLINED WAY. DON'T TRANSFER CONTROL OUT OF LOCAL AREA.
4. AVOID NULL THEN STATMEMENTS.
5. AVOID THEN_IF STATEMENTS
6. DON'T NEST TOO DEEPLY. NESTING DEEPER THAN THREE LEVELS MAKES THE LOGIC HARD TO FOLLOW.
7. HIDE DATA STRUCTURES BEHIND THE FUNCTIONS THAT ACCESS THEM.
8. USE PARENTHESES AND SPACES TO IMPROVE READABILITY.
9. DON'T USE EQUALITY TESTS ON FLOATING POINT VARIABLES.
10. DO THE VERY OBVIOUS CODE OPTIMIZATIONS AT THE BEGINNING BUT LEAVE THE INTENSIVE OPTIMIZATION EFFORT UNTIL THE CODE IS COMPLETE AND ALL AREAS CAN BE TESTED TO SEE IF RUNTIME NEEDS IMPROVED.

COMPONENT ENGINEERING PROCEDURES

INTRODUCTION

THE COMPONENT ENGINEERING ORGANIZATION PLAYS A KEY ROLE IN THE QUALITY ASSURANCE ACTIVITIES ASSOCIATED WITH SOFTWARE RELEASE AND REVISION. THE MARRIAGE OF SOFTWARE (REPRESENTED BY THE 722 NUMBER) AND HARDWARE (REPRESENTED BY THE 120 NUMBER) IS MADE IN THE PROGRAMMED CHIP (FIRMWARE) THAT HAS BEEN ASSIGNED THE 122 PART NUMBER FAMILY. EXCEPT FOR SOME SPECIAL PROCESSING AND HANDLING, A PROGRAMMED CHIP IS TREATED JUST LIKE ANY OTHER HARDWARE PART. COMPONENT ENGINEERING CONTROLS THE PRINT RELEASES FOR 120 AND 122 NUMBERS.

COMPONENT ENGINEERING HAS THE FOLLOWING RESPONSIBILITIES IN THE QUALITY ASSURANCE AND MANFACTURING RELEASE OF PRODUCT SOFTWARE:

1. THE ISSUANCE OF ALL 122-XXXX-RN PROGRAMMED MEMORY DEVICE NUMBERS. THIS INCLUDES:
   A) THE ASSIGNMENT OF A NEW 122 NUMBER WHENEVER NEEDED FOR A NEW RELEASE.
   B) THE ASSIGNMENT OF A NEW 122 NUMBER WHENEVER A CHANGE IN THE TARGET MEMORY DEVICE IS NEEDED FOR ELECTRICAL, ENVIRONMENTAL, DELIVERY, OR COST REASONS.
   C) INCREMENTING THE DASH NUMBER ON A 122 PRINT WHEN THE SOFTWARE IS REVISED.

2. ENTRY OF ALL 122-XXXX-RN NUMBERS INTO THE VAX BILL-OF-MATERIALS DATA BASE.

2. THE CREATION AND RELEASE TO PRINT CONTROL OF 122 COMPONENT SPECIFICATION CONTROL DRAWINGS.

3. ARCHIVING IN SECURE STORAGE OF MASTER PROGRAMMED MEMORY DEVICES (ALL 122 MASTERS).

4. DELIVERY OF TWO COPIES OF EACH MASTER CHIP SET TO THE GANG PROGRAMMING AREA OF RECEIVING INSPECTION.

5. PARTICIPATION IN ANY ACTIVITIES ASSOCIATED WITH CHANGING THE TARGET MEMORY DEVICE. THIS CAN BE THE CHANGE FROM ONE EPROM TO ANOTHER FOR ELECTRICAL, ENVIRONMENTAL, DELIVERY, OR COST REASONS OR THE CHANGE FROM AN EPROM TO A MASKED PART.

NEW SOFTWARE RELEASE:

TO SATISFY THE NEEDS OF PURCHASING, THE PRELIMINARY BILL-OF-MATERIAL MUST CONTAIN ALL 122 NUMBERS AND THE CORRESPONDING 122 PRINTS AND 120 PRINTS MUST ALL BE IN THE PRINT ROOM. TO ACCOMPLISH THIS, THE SOFTWARE PROJECT ENGINEER MUST IDENTIFY ALL TARGET MEMORY DEVICES AND OBTAIN PART NUMBER ASSIGNMENTS FOR SOFTWARE (722-XXXX-RN) AND FIRMWARE (122-XXXX-RN). IF THE DESIRED MEMORY DEVICE (120-XXXX-RN) DOES NOT HAVE AN ASSIGNED NUMBER, HE MUST GET THIS ALSO. THEN, THE PROJECT ENGINEER SUBMITS A REQUEST FOR A SPECIFICATION CONTROL DRAWING FOR EACH 122 NUMBER.

COMPONENT ENGINEERING THEN ASSIGNS THE 122 NUMBER (AND 120 NUMBER IF NEEDED) AND CREATES THE 122 PRINT (AND 120 PRINT IF NEEDED). THE 122 PRINT REQUIRES THE 122 NUMBER, THE 722 NUMBER, AND THE UNIT NAME WHERE THE PART WILL BE USED. SINCE PURCHASING CANNOT MAKE USE OF THE REQUIRED INFORMATION UNTIL BOTH THE 122 AND 120 NUMBERS ARE IN THE PRINT ROOM, IT IS IMPORTANT TO RELEASE THESE PRINTS AS SOON AS POSSIBLE. THE 122 PRINT SHOULD BE IN THE PRINT ROOM WITHIN A WEEK OF THE TIME THE PRINT REQUEST WAS SUBMITTED. THE 120 PRINT WILL TAKE LONGER BECAUSE OF ITS COMPLEXITY BUT SHOULD BE IN THE PRINT ROOM WITHIN THREE WEEKS FROM THE TIME OF REQUEST.

WHEN SOFTWARE CONTROL HAS CREATED THE MASTER CHIPS SETS AND THEY HAVE BEEN VERIFIED BY THE PROJECT ENGINEER, THREE COPIES OF THE CHIP SET WILL BE DELIVERED TO COMPONENT ENGINEERING. AFTER THEY ARE LOGGED IN, TWO OF THE THREE COPIES ARE DELIVERED TO THE GANG PROGRAMMING AREA OF RECEIVING INSPECTION. THE THIRD COPY IS STORED IN A SECURE AREA AND USED BY COMPONENT ENGINEERING AS A MASTER COPY TO BACKUP RECEIVING INSPECTION OR TO CLONE AND SEND AS SAMPLES TO MASKED PART VENDORS. THE PROCEDURE FOR MASTER CHIP SETS IS ALSO FOLLOWED FOR SOFTWARE REVISIONS AND CHANGES IN THE BASE ELECTRICAL PART.

SOFTWARE REVISION:

WHEN RELEASED SOFTWARE MUST BE REVISED, THE PROJECT ENGINEER WRITES AN ECO THAT CHANGES THE 122 NUMBER AND RESULTS IN A NEW LOAD FILE RELEASE BEING EXECUTED. AFTER THE REVISION LEVEL OF THE NEW 722 NUMBER HAS BEEN DETERMINED, COMPONENT ENGINEERING WILL RECIEVE A MARKED UP PRINT AND A REQUEST FROM SOFTWARE CONTROL TO CREATE A NEW 122 PRINT BY SIMPLY INCREMENTING THE DASH NUMBER OF THE PREVIOUS PRINT. FOR ONLY A SOFTWARE REVISION, THE CENTER FOUR DIGITS OF EITHER THE 122 OR 722 PART NUMBERS DO NOT CHANGE. HOWEVER, THE DASH NUMBERS FOR THE 122 AND 722 NUMBERS MUST BE ONE LEVEL HIGHER THAN THOSE ON THE PREVIOUS PRINT. FOR ANY GIVEN 122 PRINT, THE DASH NUMBERS OF THE 122 AND 722 NUMBERS MUST ALWAYS BE EQUAL.

ELECTRICAL PART CHANGE:

THERE ARE OCCASIONS WHERE A CHANGE IN THE BASE ELECTRICAL PART IS NEEDED FOR ELECTRICAL, ENVIRONMENTAL, DELIVERY, OR COST REASONS BUT THE SOFTWARE IS NOT CHANGED. IN THESE CASES, AN ECO IS WRITTEN AGAINST THE 122 PRINT REQUESTING A CHANGE TO THE NEW PART.

COMPONENT ENGINEERING WILL ASSIGN A COMPLETELY NEW 122 NUMBER (DIFFERENT CENTER DIGITS) FOR THE PROGRAMMED PART AND CREATE A NEW 122 PRINT WITH THE NEW 120 ELECTRICAL PART AND THE PREVIOUS 722 NUMBER (THE SOFTWARE HAS NOT CHANGED). THE DASH NUMBER OF THE NEW 122 NUMBER AND THE PREVIOUS 722 NUMBER MUST STILL AGREE. WHEN THE CHANGE INVOLVES GOING FROM AN EPROM PART TO A MASKED PART, THE ADDITIONAL PROCEDURES ARE NEEDED. THESE ARE DESCRIBED IN THE CHAPTERS TITLED:

"PROCEDURE FOR APPROVING EPROM CHANGE TO MASKED ROM"
"PURCHASING PROCEDURE FOR CHANGING EPROM TO MASKED ROM"

8888888888888888888888888888888888888888888888888888888888888888888888888888
30 JUNE 1983, STEVE RUSSELL

MOST OF THE FOLLOWING TEXT WAS WRITTEN BY DEBBIE KEITER.

****
PROCEDURE FOR COMPONENT ENGINEERING TO PROCESS A SOFTWARE RELEASE
****

INTRODUCTION

SOFTWARE RELEASES MAY BE EITHER NEW RELEASES OR REVISIONS TO EXISTING SOFTWARE. FOR THE NEW RELEASE, TWO FORMS ARE USED: THE 'NEW SOFTWARE: PRELIMINARY RELEASE' , AND THE 'LOAD FILE RELEASE ORDER'. FOR REVISIONS TO EXISTING SOFTWARE, THE LOAD FILE RELEASE ORDER ONLY IS USED. SOFTWARE CONTROL WILL PROVIDE COMPONENT ENGINEERING WITH A COPY OF EVERY LOAD FILE RELEASE ORDER IT RECEIVES. PLEASE PROCESS THESE ACCORDING TO SECTION A FOR NEW RELEASES; ACCORDING TO SECTION B FOR REVISIONS TO EXISTING SOFTWARE.

COMPONENT ENGINEERING WILL IN ALL CASES BE RESPONSIBLE FOR
A) ISSUING 122- NUMBERS;
B) SEEING THAT NEW 122- #S ARE PUT ON VAX;
C) CREATING 122- # PIECE PART DRAWINGS AND RELEASING THEM TO BLUEPRINT;
D) HOLDING MASTER 122- # MICROPROCESSOR CHIPS IN SAFE STORAGE.

A: NEW RELEASES

NEW RELEASES (FOR 122- NUMBERS ENDING IN -00) WILL BE EFFECTED IN TWO STAGES.
THE FIRST STAGE OCCURS AT THE SAME TIME AS THE PURCHASING (INCLUDING THE BILL OF MATERIAL) RELEASE. AT THIS TIME ALL APPROPRIATE PART NUMBERS WILL BE ASSIGNED AND A 122- NUMBER PIECE PARTS PRINT WILL BE GENERATED.
THE SECOND STAGE OCCURS AT THE SAME TIME AS THE MANUFACTURING RELEASE; FOR COMPONENT ENGINEERING, THIS STAGE WILL PRIMARILY BE A DOUBLE-CHECK OF INFORMATION OBTAINED DURING THE PRELIMINARY (STAGE 1) RELEASE.

STAGE 1: NEW SOFTWARE: PRELIMINARY RELEASE

THE SOFTWARE PROJECT ENGINEER WILL PARTIALLY FILL OUT THE FORM ENTITLED 'NEW SOFTWARE: PRELIMINARY RELEASE'. HE WILL OBTAIN DOCUMENT NUMBERS FROM THE ENGINEERING OFFICE (JAN ODELL). HE WILL THEN BRING THE FORM TO COMPONENT ENGINEERING.

COMPONENT ENGINEERING WILL ASSIGN THE 122- NUMBER OR NUMBERS AND WILL MAKE A COPY OF THE FORM. THE ENGINEER WILL KEEP THE ORIGINAL FOR FUTURE REFERENCE. COMPONENT ENGINEERING WILL SEE THAT THE 122- NUMBER AND STANDARD DESCRIPTION ARE PUT ON VAX. COMPONENTS WILL ALSO GENERATE A PIECE PARTS PRINT FOR THE 122- NUMBER OR NUMBERS, USING THE INFORMATION ON THE PRELIMINARY RELEASE FORM, AND WILL PLACE THIS PRINT IN BLUEPRINT. THIS WILL THEN BE AVAILABLE TO PURCHASING DURING THEIR ASSESSMENT OF PIECE PART ORDERS CHANGED OR MADE NECESSARY BY THE PROJECT.

INFORMATION AS TO WHETHER A PART WILL EVENTUALLY BE AVAILABLE FOR MASKING WILL NO LONGER BE INCLUDED ON THE SPECIFICATION DRAWING SINCE INITIAL MASKING REQUESTS WILL COME FROM PURCHASING, BASED ON THE USE OF A LARGE NUMBER OF PROGRAMMED DEVICES.

WHEN COMPONENT ENGINEERING SEES THE 'NEW SOFTWARE: PRELIMINARY RELEASE' DOCUMENT, ALL INFORMATION EXCEPT THE 122- NUMBER SHOULD ALREADY BE FILLED IN. IF IT IS NOT, THE OMISSIONS SHOULD BE POINTED OUT TO THE PROJECT ENGINEER; IF HE HAS QUESTIONS, HE SHOULD BE REFERRED TO SOFTWARE CONTROL.

STAGE 2: LOAD FILE RELEASE ORDER

AT THE TIME OF THE MANUFACTURING RELEASE, THE SOFTWARE PROJECT ENGINEER WILL FILL OUT A LOAD FILE RELEASE ORDER. THIS WILL PRIMARILY CONCERN SOFTWARE CONTROL, BUT COMPONENT ENGINEERING WILL RECEIVE A COPY OF THE RELEASE ORDER AND SHOULD CHECK THAT THE INFORMATION ON THE RELEASE ORDER CORRESPONDS WITH COMPONENTS' RECORDS AS TO:

A) 122- NUMBERS AND STANDARD DESCRIPTIONS
B) INFORMATION CONTAINED ON THE 122- NUMBER PIECE PARTS PRINT.

COMPONENTS SHOULD ALWAYS BE CAREFUL TO NOTE THE LAST TWO DIGITS OF ANY LOAD FILE RELEASE ORDER (720- NUMBER) IT RECEIVES. IF THESE ARE -00, THE DOCUMENT CONSTITUTES STAGE 2 OF A NEW SOFTWARE RELEASE AND SHOULD BE TREATED AS DESCRIBED HERE. IF, HOWEVER, THE LAST TWO DIGITS OF THE 720- NUMBER ARE ANYTHING OTHER THAN -00, THE RELEASE ORDER SHOULD BE TREATED AS DESCRIBED IN SECTION B BELOW.

B: REVISIONS TO EXISTING SOFTWARE

IN CASES WHERE A REVISION IS MADE TO EXISTING SOFTWARE, THE LOAD FILE RELEASE ORDER SHALL BE COMPLETED AND SUBMITTED WITH THE ECO WHICH CHANGES THE 122- NUMBER ON THE PURCHASING BILL OF MATERIAL. THE ECO DEPARTMENT WILL FORWARD THE LOAD FILE RELEASE ORDER TO SOFTWARE CONTROL; SOFTWARE CONTROL WILL THEN FORWARD A COPY TO COMPONENT ENGINEERING.

REMEMBER THAT SOFTWARE REVISIONS ARE EFFECTED BY ASSIGNING A NEW 122- NUMBER WHICH INCREMENTS THE LAST TWO DIGITS OF THE PREVIOUS 122- NUMBER, NOT BY ISSUING A REVISION LEVEL TO THE EXISTING NUMBER AND PRINT. THIS MEANS THAT THE 122- NUMBER FOR A SOFTWARE REVISION IS PREDETERMINED: THE NEW NUMBER WILL ALWAYS BE IDENTICAL TO THE OLD NUMBER, EXCEPT THAT THE LAST TWO DIGITS WILL BE INCREMENTED BY ONE. FOR EXAMPLE, IF THE EXISTING NUMBER IS 122-0038-01, AND THE ENGINEER IS WRITING A REVISION TO THE SOFTWARE, HE KNOWS THAT THE NEW 122- NUMBER WILL AUTOMATICALLY BE 122-0038-02.

WHEN YOU RECEIVE A LOAD FILE RELEASE ORDER WHICH REVISES EXISTING SOFTWARE (IF THE 720- NUMBER ENDS IN ANYTHING BUT -00, THEN THE DOCUMENT IS REVISING EXISTING SOFTWARE), PLEASE PROCEED AS FOLLOWS:

1. IN THE EVEN-BYTE AND ODD-BYTE SECTIONS ON PAGE 1, LOOK FOR CHECK MARKS IN THE COLUMNS MARKED 'OLD' AND 'NEW' NEXT TO THE 122-# SLOTS.

ANY CHECK MARKS IN THE 'OLD' COLUMN MAY BE IGNORED: THESE ARE NUMBERS WHICH HAVE ALREADY BEEN ASSIGNED, PUT ON VAX, AND HAD PRINTS MADE FOR A PREVIOUS RELEASE.

A CHECK IN THE 'NEW' COLUMN INDICATES THAT THE NUMBER HAS NOT BEEN PREVIOUSLY DEFINED. THIS MEANS THAT ALTHOUGH THE NUMBER IS PREDETERMINED, IT MUST BE OFFICIALLY 'ASSIGNED' AND PUT ON VAX.

2. NOTE ANY NEW NUMBERS (TOGETHER WITH THEIR STANDARD DESCRIPTIONS) IN YOUR RECORDS AS BEING ASSIGNED.

AT THE PRESENT TIME, COMPONENT ENGINEERING NEED NOT BE RESPONSIBLE FOR SEEING THAT THESE NEW NUMBERS IS PUT ON VAX, SINCE THE APPROPRIATE PERSONNEL (JAN ODELL) WILL ALSO BE RECEIVING A COPY OF THE LOAD FILE RELEASE ORDER AND WILL BE ENTERING NEW NUMBERS AND THEIR STANDARD DESCRIPTIONS ON VAX.

3. A CHECK MARK IN THE 'NEW' COLUMN ALSO CONSTITUTES A REQUEST THAT A SPECIFICATION CONTROL DRAWING BE GENERATED FOR THAT 122-#. COMPONENT ENGINEERING IS RESPONSIBLE FOR SEEING THAT SUCH A PRINT IS GENERATED AND DELIVERED TO BLUEPRINT. MASTER COPIES OF MICROPROCESSOR CHIPS WILL NOT BE DELIVERED TO COMPONENT ENGINEERING UNTIL THE APPROPRIATE 122- PRINT IS AVAILABLE. ALL THE INFORMATION NEEDED TO GENERATE THE PRINT IS CONTAINED ON

THE LOAD FILE RELEASE ORDER AS FOLLOWS:

A) THE 120- NUMBER IS DIRECTLY BELOW THE NEW 122- NUMBER ON THE LOAD FILE RELEASE ORDER. COMPONENTS IS RESPONSIBLE FOR CHECKING THAT THIS IS A VALID KING PART NUMBER, AND THAT A SPECIFICATION CONTROL DRAWING IS IN PROCESS IF NOT COMPLETED AND AVAILABLE FROM BLUEPRINT.

B) THE 722- NUMBER IS DIRECTLY BELOW THE 120- NUMBER ON THE LOAD FILE RELEASE ORDER.

C) THE EXPECTED RELEASE DATE OF THE BINARY IMAGE FILE (722- NUMBER) IS FOUND IN THE 'CHIP #' SECTION ON THE SECOND PAGE.

D) THE UNIT OR UNITS IN WHICH THE MICROPROCESSOR IS USED ARE NAMED AT THE TOP OF PAGE 1, TO THE RIGHT OF THE 'SYSTEM NAME (S)' SECTION.

IT SHOULD TAKE APPROXIMATELY THE SAME AMOUNT OF TIME FOR COMPONENT ENGINEERING TO PROCESS A PRINT REQUEST, HAVE IT COMPLETED, AND DELIVER IT TO BLUEPRINT AS THE AVERAGE ENGINEERING CHANGE ORDER TURNOVER TIME. THIS WILL ENSURE THAT THE COMPLETED 122- PRINT WILL ARRIVE IN BLUEPRINT AT THE SAME TIME AS THE REVISED BILL OF MATERIALS.

C. HANDLING OF MASTER CHIPS

AT THE CONCLUSION OF A LOAD FILE RELEASE, SOFTWARE CONTROL WILL HAVE CREATED FOUR MASTER CHIPS FOR EACH 122- NUMBER DEFINED ON THE LOAD FILE RELEASE ORDER. THESE CHIPS WILL BE LABELED WITH THE APPROPRIATE 122- NUMBER. ONE OF THE FOUR CHIPS WILL BE RETURNED TO THE SOFTWARE PROJECT ENGINEER FOR HIS RECORDS; THE OTHER THREE MASTER CHIPS WILL BE DELIVERED TO COMPONENT ENGINEERING.

COMPONENTS IS RESPONSIBLE FOR LOGGING IN THE RECEIPT OF THE CHIPS, TOGETHER WITH THEIR 122- NUMBER AND THE DATE THEY WERE RECEIVED; FOR STORING ONE OF THE CHIPS PERMANENTLY IN SUCH A PLACE AND MANNER THAT IT SHALL NOT BE SUBJECT TO CONTAMINATION OR TO MAGNETIC OR ELECTRICAL INTERFERENCE; AND FOR KEEPING THE REMAINING TWO CHIPS SAFELY UNTIL THEY ARE DELIVERED TO A REPRESENTATIVE FROM RECEIVING INSPECTION.

** 7.0 SOFTWARE CONTROL LIBRARY PROCEDURES

** 7.1 INTRODUCTION

A KEY ELEMENT OF EFFECTIVE SOFTWARE CONFIGURATION MANAGEMENT IS THE SOFTWARE CONTROL LIBRARY. THE SOFTWARE CONTROL LIBRARY CONTAINS THE DOCUMENTATION OF ALL OF THE SOFTWARE RELEASES AND THE ARCHIVED FILES.

LIBRARY CONTENTS:

1) STAGE-1 RELEASE FORM FOR TOP-LEVEL SOFTWARE DOCUMENTS.
2) STAGE-2 RELEASE FORM FOR LRU SOFTWARE DOCUMENTS.
3) STAGE-3 RELEASE FORM FOR THE LOAD FILE DOCUMENT.
4) SOURCE AND COMMAND FILES FOR NEWLY RELEASED SOFTWARE.
5) PRIMARY BACKUP OF SOURCE AND COMMAND FILES FOR EACH LOAD FILE
6) REMOTE (SECONDARY) BACKUP FILES.

LIBRARIAN DUTIES:

1) MAINTAIN AND REVISE RELEASE FORMS.
2) MAINTAIN LIBRARY FILE STRUCTURE ON VAX FOR RELEASED TARGET SOFTWARE. CREATE NEW SUBDIRECTORIES WITH NAME CORRESPONDING TO THE LOAD FILE NUMBER.
3) RUN THE SUPPORT SOFTWARE COMMAND PROCEDURE TO CREATE NECESSARY DOCUMENT FILES AND ASCII-HEX FILES.
4) PROGRAM FOUR MASTER CHIPS FOR EACH ASCII-HEX FILE TO BE USED IN THE VERIFICATION PROCESS AND THEN GIVEN TO COMPONENT ENGINEERING.
5) CREATE AND RELEASE THE LOAD FILE DOCUMENT.
6) CREATE AND RELEASE THE BINARY IMAGE FILE DOCUMENT.
7) MAINTAIN PROCEDURAL DOCUMENTS.

NAME: STEVE RUSSELL [SOFTLIB]SCMGUIDE.LIS
DATE: 23 DEC 1982

SOFTWARE CONFIGURATION MANAGEMENT PROCEDURE FOR A LOAD FILE SOFTWARE RELEASE AND THE CREATION OF MASTER CHIPS FOR PRODUCTION.

FROM: STEVE RUSSELL
DATE: 23 DEC 1982

---

1. ASSIST THE SOFTWARE ENGINEER IN COMPLETING THE LOAD FILE RELEASE FORM (799-0003-00). MAKE SURE THEY HAVE A COPY OF THE RELEASE PROCEDURE AND HAVE READ IT PRIOR TO SIGNING THE AUTHORIZATION FOR A 722 NUMBER ASSIGNMENT.

2. AFTER RECEIVING A COMPLETED SOFTWARE DOCUMENT NUMBER REQUEST FORM, CHECK IT FOR COMPLETENESS OF INFORMATION AND RETURN IT TO INITIATOR FOR ADDITIONAL INFORMATION IF NECESSARY.

3. CHECK TO SEE IF THE SPECIFIED 122 PRINT IS AVAILABLE IN THE PRINT ROOM. IF NOT MAKE SURE THE INITIATOR HAS APPLIED FOR IT TO COMPONENT ENGINEERING. DO NOT DISTRIBUTE MASTER CHIPS UNTIL THE PRINT IS AVAILABLE! THE DESIGNER SHOULD SEE THAT THE 122 PRINT IS AVAILABLE AT THE TIME OF PURCHASING RELEASE SO PURCHASING CAN PULL IT AND MAKE PLANS TO HAVE THE UNPROGRAMMED PARTS AVAILABLE WHEN NEEDED.

4. ASK THE DESIGNER TO PUT A COPY OF THE SOURCE FILE AND .COM FILE SPECIFIED ON THE RELEASE FORM INTO THE [EXCHANGE] DIRECTORY. THEN CREATE A NEW .720XXXXRN SUBDIRECTORY UNDER THE [SOFTLIB] DIRECTORY AND COPY THESE FILES INTO IT.

5. REVIEW THE SOURCE FILE TO MAKE SURE IT HAS A COMPLETE AND DESCRIPTIVE HEADER OF INFORMATION. ALSO REVIEW THE .COM FILE TO SEE THAT IT MEETS REQUIREMENTS OF FORM AND THAT THE SPECIFIED EXECUTABLES AGREE WITH THOSE ON THE LOAD FILE RELEASE FORM. CONSULT WITH THE DESIGNER ABOUT ANY DIFFERENCES AND MAKE NECESSARY CHANGES.

6. RUN THE COMMAND PROCEDURE AND

3. AFTER THE 722 NUMBER HAS BEEN ASSIGNED, APPLY TO COMPONENT ENGINEERING (P.J. OR DALE COOPER) FOR A PROGRAMMED DEVICE PART NUMBER 122-XXXX-RN. THIS IS THE NUMBER THAT WILL BE PUT ON THE BILL-OF-MATERIALS. YOU

WILL BE REQUIRED TO FILL OUT A FORM GIVING THE 722 NUMBER, THE GENERIC PART NUMBER (120-XXXX-XX) OF THE ELECTRICAL PART TO BE PROGRAMMED, AND THE EXPECTED RELEASE DATE OF THE FINISHED SOFTWARE.

4. AFTER THE 122 NUMBER HAS BEEN ASSIGNED, YOU WILL BE ABLE TO FILL OUT THE LOAD FILE RELEASE FORM OBTAINED IN STEP #1.
   A) ENTER THE LOAD FILE SOFTWARE DOCUMENT NUMBER (720) AT THE TOP AND BOTTOM OF EACH PAGE.
   B) ENTER THE SYSTEM NAMES OF ALL SYSTEMS THAT WILL BE USING THIS PROGRAMMED DEVICE. GENERALLY ONLY ONE SYSTEM NAME WILL BE ENTERED.
   C) ENTER THE NAMES OF ALL UNITS (LRUS) THAT WILL BE USING THE PROGRAMMED DEVICE. FOR EACH DIFFERENT UNIT, A DIFFERENT 715 NUMBER IS REQUIRED UNLESS ALL OF THE SOFTWARE IN EACH DIFFERENT UNIT IS IDENTICAL.
   D) ENTER THE NAME OF THE PROCESSOR THAT WILL BE USING THE SOFTWARE DESCRIBED BY THIS LOAD FILE DOCUMENT. BE SURE TO USE A UNIQUE AND DESCRIPTIVE NAME.
   E) ENTER THE VAX FILE NAMES FOR THE COMMAND FILE, LOAD FILE, SOURCE FILE, AND OBJECT FILE. TRY TO USE A UNIQUE AND DESCRIPTIVE NAME AND ALSO PROVIDE FOR A METHOD OF NOTING THE REVISION LEVEL OF THE SOURCE. FOR EXAMPLE, THE NAMES FOR THE KAC-952 ANTENNA COUPLER PROCESSOR COULD BE KAC952V01.COM, KAC952V01.LOA, KAC952V01.I48, KAC952V01.LOA. IN THIS CASE, THE SOURCE FILE IS .I48 AND THE VERSION LEVEL (V01) IS PART OF THE VAX FILENAME.
   F) ENTER THE NAMES OF THE SOFTWARE PROJECT ENGINEER AND THE ENGINEERING GROUP LEADER. THESE ARE THE PEOPLE THAT WILL AUTHORIZE THE ACTUAL RELEASE OF MASTER CHIPS TO MANUFACTURING.
   G) ENTER THE PROCESSOR SYSTEM NAME AGAIN ON THE BOTTOM LINE OF THE FIRST PAGE.
   H) ENTER THE ASSIGNED PART NUMBERS ON THE SECOND PAGE WHERE INDICATED. IF THE RELEASE IS FOR A 16-BIT PROCESSOR, THE EVEN AND ODD BYTE VAX FILE NAMES MUST ALSO BE ASSIGNED. THE LAST SIX DIGITS OF THE 722 DOCUMENT NUMBER ARE ALSO USED IN THE DOCUMENT FILE NAME AND ASCII-HEX FILE NAME.
   I) INDICATE THE PROGRAMMED DEVICE TYPE IN THE BLANK PROVIDED. THIS IS USED BY SOFTWARE CONTROL AS A CROSSCHECK TO MAKE SURE THE CORRECT TARGET DEVICE IS USED TO MAKE MASTER COPIES.
   J) INDICATE THE SUPPORT SOFTWARE VERSIONS THAT YOU WANT USED IN THE RELEASE. THESE SHOULD AGREE WITH THE VERSIONS THAT ARE SPECIFIED IN THE .COM FILE REFERENCED ON THE FIRST PAGE.

5. AFTER THE FORM HAS BEEN FILLED OUT, RETURN IT TO SOFTWARE CONTROL. THE INFORMATION PROVIDED ON THE LOAD RELEASE FORM WILL NOW BE USED TO CREATE THE NECESSARY LOAD FILES AND DOCUMENT FILES ON VAX. THE PROCEDURE USED INSURES THAT ALL SOFTWARE AND SUPPORT DOCUMENTS WILL BE ARCHIVED AND THAT THE LOAD FILES CAN BE RECREATED IN THE FUTURE WITH THE IDENTICAL SOURCE FILE AND SOFTWARE SUPPORT ENVIRONMENT.

6. THE SOURCE AND LOAD FILES WILL BE PLACED IN THE [EXCHANGE] DIRECTORY AND YOU WILL BE ASKED TO DO A DIRECT COMPARISION OF THE CODE GENERATED IN THE RELEASE WITH THE CODE YOU ARE USING IN THE CURRENT ENGINEERING MODEL. THIS STEP ELIMINATES OBVIOUS OR GROSS ERRORS PRIOR TO THE CREATION OF MASTER CHIPS.

7. AFTER YOU HAVE VERIFIED THAT THE CODE IS CORRECT ON VAX, SOFTWARE CONTROL WILL AGAIN USE THE INFORMATION PROVIDED ON THE LOAD RELEASE FORM TO CREATE FOUR MASTER COPIES OF THE PROGRAMMED DEVICE. THESE MASTERS ARE USED BY MANUFACTURING TO GANG PROGRAM PRODUCTION CHIPS. THEY ARE ALSO USED IN QUALITY CONTROL PROCEDURES SUCH AS THE VERIFICATION AND VALIDATION OF THE PROGRAMMED DEVICE BY THE SOFTWARE PROJECT ENGINEER. SOMETIMES YOU MAY BE ASKED TO PROVIDE ERASED DEVICES FOR THIS PURPOSE.

8. AFTER A MASTER CHIP SET HAS BEEN PRODUCED, A SINGLE COPY WILL BE RETURNED TO THE DESIGN GROUP FOR VERIFICATION AND VALIDATION. THIS INVOLVES A BIT-BY-BIT VERIFICATION OF THE MACHINE CODE IN THE MASTER CHIP AGAINST THE CORRESPONDING CHIP IN THE ENGINEERING PROTOTYPE. THIS IS USUALLY DONE ON A PROM PROGRAMMER WITH A VERIFY OR COMPARE COMMAND. IN ADDITION, THE MASTER CHIP SHOULD BE USED IN THE PROTOTYPE TO VALIDATE ALL SOFTWARE FUNCTIONS IMPLEMENTED WITH THIS PARTICULAR CHIP.

9. AFTER VERIFICATION AND VALIDATION, THE SOFTWARE PROJECT ENGINEER AND THE ENGINEERING GROUP LEADER WILL SIGN AND DATE THE LOAD FILE RELEASE FORM. THIS SIGNIFIES THAT THE SOFTWARE IS CORRECT SO IT CAN BE ARCHIVED ON MAGNETIC TAPE AND RELEASED TO MANUFACTURING. THE DESIGN GROUP KEEPS THEIR MASTER COPY AS A READY REFERENCE TO THE ACTUAL CODE IN THE RELEASED CHIP. AT THIS TIME, SOFTWARE CONTROL WILL ARCHIVE ALL VAX FILES ASSOCIATED WITH THIS RELEASE AND WILL SUBMIT THE REMAINING THREE MASTER COPIES TO COMPONENT ENGINEERING.

ARCHIVING AND RESTORING PROCEDURES FOR PRODUCT SOFTWARE

ALL PRODUCT SOFTWARE FILES AND THEIR ASSOCIATED DIRECTORY STRUCTURE ARE STORED ON MAGNETIC MEDIA SO THAT THEY CAN BE RESTORED TO DO REVISIONS. THERE ARE TWO TYPES OF ARCHIVING PROCEDURES USED FOR PRODUCT SOFTWARE. THE FIRST, REFERRED TO AS AN INCREMENTAL BACKUP, IS DONE FOR NEW RELEASES. THE SECOND, REFERRED TO A PERMANENT BACKUP, IS DONE WHEN MANY RELEASES HAVE ACCUMULATED ON DISK AND NEED TO BE TRANSFERRED PERMANENTLY TO MAGNETIC TAPE. EACH COLLECTION OF PRODUCT SOFTWARE FILES IS ALWAYS BACKED UP IN AT LEAST TWO SEPARATE LOCATIONS.

INCREMENTAL BACKUPS

FIRST, WE WILL DESCRIBE THE PROCEDURE FOR DOING AN INCREMENTAL BACKUP OF A NEW RELEASE. NEW RELEASES ARE STORED BOTH ON DISK AND ON A TAPE VOLUME NAMED SCM001. THE PROCEDURE IS DESCRIBED IN THE FOLLOWING STEPS:

1. SIGN ON VAX IN THE SOFTLIB ACCOUNT AND ALLOCATE THE TAPE DRIVE USING THE DCL COMMAND:

   $ ALLOCATE MTA0:

   THIS RESERVES THE TAPE STAND FOR EXCLUSIVE USE BY SOFTLIB AND IS NEEDED TO PROTECT THE TAPE VOLUME FROM ACCIDENTAL OVERWRITE BY ANOTHER USER. MTA0 IS THE DEVICE NAME FOR MAGNETIC TAPE DRIVE A0.

2. IF THE TERMINAL IS SHARED BY OTHER USERS, LEAVE A NOTE ON THE TERMINAL STATING THAT IT IS BEING USED FOR MAG TAPE BACKUP. IF SOMEONE WERE TO LOG SOFTLIB OFF, THE TAPE WOULD BE DISMOUNTED AND THE DRIVE DEALLOCATED.

3. GO TO THE VAX COMPUTER ROOM AND GET THE SOFTWARE CONTROL ARCHIVE TAPE LABELED SCM001 FROM THE TAPE CABINET. IF THE CABINET IS LOCKED HAVE ONE OF THE COMPUTER SYSTEM PERSONNEL UNLOCK IT.

4. VOLUME SCM001 WILL HAVE A WRITE RING ON IT SO ONE WILL NOT HAVE TO BE INSTALLED. SINCE IT IS USED ONLY FOR TEMPORARY BACKUP AND THE FILES STILL ARE ON DISK, IT WOULD NOT BE CATASTROPHIC IF THE VOLUME WAS ACCIDENTLY OVERWRITTEN. MOUNT THE TAPE VOLUME ON THE TAPE DRIVE CONNECTED TO THE SILVER SYSTEM.

5. PRESS THE LOAD/REW BUTTON TO ELECTRICALLY LOAD THE TAPE. WHEN THE LOAD IS COMPLETE, PRESS THE ON LINE BUTTON. THE TAPE IS NOW READY TO BE LOGICALLY MOUNTED.

6. RETURN TO THE SOFTLIB TERMINAL AND MOUNT THE TAPE LOGICALLY USING THE DCL COMMAND:

   $ MOUNT/FOREIGN MTA0:

   THE TAPE IS MOUNTED /FOREIGN SO THAT THE BACKUP UTILITY CAN BE USED. BACKUP WILL NOT WORK WHEN A TAPE IS MOUNTED WITH A FILES-11 STRUCTURE.

7. USE THE DCL BACKUP COMMAND TO CREATE A CONTAINER FILE (SAVE-SET). AN EXAMPLE OF THE PROPER COMMAND SYNTAX IS:

   $ BACKUP/NOREWIND/LOG [SOFTLIB.720000200] MTA0:720000200.BCK

   WITH THIS COMMAND, ALL FILES IN SUBDIRECTORY [SOFTLIB.720000200] ARE WRITTEN TO A SAVE-SET NAMED 720000200.BCK ON THE TAPE. IT IS VITAL THAT THE /NOREWIND QUALIFIER BE USED SO THAT THE SAVE-SET WILL BE WRITTEN FOLLOWING THE END OF THE LAST SAVE-SET AND NOT AT THE BEGINNING OF THE TAPE. THIS PREVENTS WRITING OVER PREVIOUS SAVE-SETS. THE ONLY TIME THAT THE /NOREWIND QUALIFIER SHOULD NOT BE USED IS WHEN ALL FILES ON VOLUME SCM001 HAVE BEEN PERMENANTLY BACKED UP ON VOLUMES SCM01A AND SCM01B AND SOFTWARE CONTROL IS STARTING A NEW SERIES OF INCREMENTAL BACKUPS. NOTICE THAT THE SAVE-SET NAME IS THE SAME AS THE SUBDIRECTORY NAME FOR THE LOAD FILE. THE /LOG QUALIFIER CAUSES A LIST OF THE FILES TRANSFERRED TO THE SAVE-SET TO BE PRINTED ON THE TERMINAL. THIS IS USED AS EXTRA VERIFICATION THAT THE PROPER FILES WERE INDEED WRITTEN TO THE TAPE.

8. DISMOUNT THE TAPE VOLUME USING THE DCL COMMAND:

   $ DISMOUNT MTA0:

   THIS COMMAND WILL REWIND THE TAPE, TAKE IT OFF-LINE, AND PHYSICALLY UNLOAD IT FROM THE TAPE DRIVE.

9. RETURN TO THE VAX COMPUTER ROOM, REMOVE THE VOLUME FROM THE DRIVE AND RETURN IT TO THE TAPE CABINET.

10. RETURN TO THE SOFTLIB TERMINAL AND ENTER THE DCL COMMAND:

    $ DEALLOCATE MTA0:

    THIS WILL DEALLOCATE THE TAPE DRIVE MTA0 SO THAT IT MAY BE USED BY OTHERS.

PERMANENT BACKUP

THE PROCEDURES FOR PERMANENT BACKUP ARE VERY SIMILIAR TO THOSE USED FOR INCREMENTAL BACKUP EXCEPT THAT THE PRODUCT SOFTWARE FILES ARE WRITTEN TO TWO VOLUMES INSTEAD OF JUST ONE AND THE FILES ON DISK ARE DELETED WHEN THE BACKUP IS COMPLETED. THE PROCEDURE IS AS FOLLOWS:

1. SIGN ON VAX IN THE SOFTLIB ACCOUNT AND ALLOCATE THE TAPE DRIVE USING THE DCL COMMAND:

   $ ALLOCATE MTA0:

   THIS RESERVES THE TAPE STAND FOR EXCLUSIVE USE BY SOFTLIB AND IS NEEDED TO PROTECT THE TAPE VOLUME FROM ACCIDENTAL OVERWRITE BY ANOTHER USER. MTA0 IS THE DEVICE NAME FOR MAGNETIC TAPE DRIVE A0.

2. IF THE TERMINAL IS SHARED BY OTHER USERS, LEAVE A NOTE ON THE TERMINAL STATING THAT IT IS BEING USED FOR MAG TAPE BACKUP. IF SOMEONE WERE TO LOG SOFTLIB OFF, THE TAPE WOULD BE DISMOUNTED AND THE DRIVE DEALLOCATED.

3. GO TO THE VAX COMPUTER ROOM AND GET THE SOFTWARE CONTROL ARCHIVE TAPE LABELED SCM01B FROM THE TAPE CABINET. IF THE CABINET IS LOCKED HAVE ONE OF THE COMPUTER SYSTEM PERSONNEL UNLOCK IT. VOLUME SCM01A IS IN THE SAFE. CONTACT THE VAX SYSTEM MANAGER FOR ASSISTANCE IN GETTING IT.

4. NEITHER VOLUME WILL HAVE A WRITE RING ON IT SO ONE WILL HAVE TO BE INSTALLED. SINCE THESE ARE PERMANENT COPIES, IT WOULD BE CATASTROPHIC IF BOTH VOLUMES WERE ACCIDENTALLY OVERWRITTEN WHILE TRYING TO RESTORE PRODUCT SOFTWARE. FOR THIS REASON, THE WRITE RINGS ARE LEFT OFF EXCEPT DURING A PERMANENT BACKUP.

THE FOLLOWING STEPS APPLY TO BOTH VOLUMES ALTHOUGH THE EXAMPLES WILL REFER TO VOLUME SCM01A. REPEAT THE STEPS FOR VOLUME SCM01B.

5. MOUNT THE TAPE VOLUME ON THE TAPE DRIVE CONNECTED TO THE SILVER SYSTEM.

6. PRESS THE LOAD/REW BUTTON TO ELECTRICALLY LOAD THE TAPE. WHEN THE LOAD IS COMPLETE, PRESS THE ON LINE BUTTON. THE TAPE IS NOW READY TO BE LOGICALLY MOUNTED.

7. RETURN TO THE SOFTLIB TERMINAL AND MOUNT THE TAPE LOGICALLY USING THE DCL COMMAND:

   $ MOUNT/FOREIGN MTA0:

   THE TAPE IS MOUNTED /FOREIGN SO THAT THE BACKUP UTILITY CAN BE USED. BACKUP WILL NOT WORK WHEN A TAPE IS MOUNTED WITH A FILES-11 STRUCTURE.

8. CREATE A COMMAND PROCEDURE NAMED SCM01.COM (OR EDIT A PREVIOUS ONE) THAT WILL COPY (USING BACKUP) ALL DESIRED FILES AND THEIR ASSOCIATED DIRECTORY STRUCTURE INTO THE DESIRED CONTAINER FILES. IF A PREVIOUS COMMAND PROCEDURE IS EDITED, BE SURE TO REMOVE ALL BACKUP COMMANDS FOR PREVIOUS FILES. AN EXAMPLE OF THIS TYPE OF COMMAND PROCEDURE IS:

```
$! SILVER::ENG:[SOFTLIB]SCM01.COM
$! THIS COMMAND PROCEDURE IS USED TO BACKUP ARCHIVE VOLUMES
$! SCM01A AND SCM01B.
$! DATE:                    NAME:
$!
```

```
$ BACKUP/NOREWIND/LOG  [SOFTLIB]SCM01.COM    MTA0:SCM01.COM
$ BACKUP/NOREWIND/LOG  [SOFTLIB.7200023001]  MTA0:720002300.BCK
$ BACKUP/NOREWIND/LOG  [SOFTLIB.7200024001]  MTA0:720002400.BCK
$ BACKUP/NOREWIND/LOG  [SOFTLIB.7200025001]  MTA0:720002500.BCK
$ BACKUP/NOREWIND/LOG  [SOFTLIB.7200026001]  MTA0:720002600.BCK
$ BACKUP/NOREWIND/LOG  [SOFTLIB.7200027001]  MTA0:720002700.BCK
$ BACKUP/NOREWIND/LOG  [SOFTLIB.7200029001]  MTA0:720002900.BCK
$ BACKUP/NOREWIND/LOG  [SOFTLIB.7200031001]  MTA0:720003100.BCK
$    EXIT
```

IT IS VITAL THAT THE /NOREWIND QUALIFIER ALWAYS BE USED! IF NOT, THE NEW SAVE-SETS WILL DESTROY THE OLD ONES BY WRITING OVER THEM. NOTICE THAT THE SAVE-SET NAME IS THE SAME AS THE SUBDIRECTORY NAME FOR THE LOAD FILE. THE /LOG QUALIFIER CAUSES A LIST OF THE FILES TRANSFERRED TO THE SAVE-SET TO BE PRINTED ON THE TERMINAL. THIS IS USED AS EXTRA VERIFICATION THAT THE PROPER FILES WERE INDEED WRITTEN TO THE TAPE.

9. SET UP THE SOFTLIB TERMINAL AND ASSOCIATED PRINTER SO THAT ALL OF THE DCL COMMANDS THAT FOLLOW WILL BE LOGGED.

10. RUN THE COMMAND PROCEDURE WITH THE DCL COMMAND:

    $ @SCM01.COM

    EACH BACKUP COMMAND IN SCM01.COM WILL BE EXECUTED AND THE RESULTING LOG PRINTED.

11. ENTER THE FOLLOWING DCL COMMANDS:

    ```
    $ DISMOUNT/NOUNLOAD MTA0:
    $ MOUNT MTA0: SCM01A
    $ DIRECTORY MTA0:
    ```

    THESE WILL REWIND THE TAPE WITHOUT UNLOADING IT, LOGICALLY MOUNT IT WITH A FILES-11 STRUCTURE, AND LIST ALL OF THE CONTAINER FILE NAMES. THE SOFTLIB PRINTER WILL BE RECORDING THESE FOR DOCUMENTATION.

12. ENTER THE FOLLOWING DCL COMMANDS:

    ```
    $ TIME
    $ SHOW PROCESS
    ```

    THESE WILL LIST THE TIME AND DATE THE PERMANENT BACKUP OCCURED AND ALL OF THE INFORMATION ABOUT THE SOFTLIB PROCESS VALID WHEN THE BACKUP OCCURED. THIS INFORMATION WOULD BE USED IN CASES WHERE ERRORS OR PROBLEMS OCCURED WITH THE BACKUP.

13. DISMOUNT THE TAPE VOLUME USING THE DCL COMMAND:

    $ DISMOUNT MTA0:

    THIS COMMAND WILL REWIND THE TAPE, TAKE IT OFF-LINE, AND PHYSICALLY UNLOAD IT FROM THE TAPE DRIVE.

14. REMOVE THE PRINTOUT FROM THE SOFTLIB PRINTER AND RETURN TO THE VAX COMPUTER ROOM. REMOVE THE VOLUME FROM THE DRIVE. REMOVE THE WRITE RING AND RETURN THE VOLUME TO ITS PROPER STORAGE ALONG WITH THE PRINTOUT.

15. REPEAT STEPS 5 THOUGHT 14 FOR VOLUME SCM018.

16. RETURN TO THE SOFTLIB TERMINAL AND ENTER THE DCL COMMAND:

    $ DEALLOCATE MTA0:

    THIS WILL DEALLOCATE THE TAPE DRIVE MTA0 SO THAT IT MAY BE USED BY OTHERS.

17. FILE THE PREVIOUS PRINTOUTS FOR BOTH SCM01A AND SCM01B IN THE SOFTWARE CONTROL FILE IN THE FOLDER LABELED SCM01A AND SCM01B HARDCOPY PRINTS.

THIS COMPLETES THE PROCEDURE FOR PERMANENT BACKUP OF NEW RELEASES.

RESTORING FILES

GET VOLUME
COPY TO DISK BOTH DIRECTORY STRUCTURE AND FILES
PUT FILES IN EXCHANGE FOR DESIGNER.

**888

EXPLAIN HOW TO RESTORE
TO RESTORE THE DIRECTORY YOU

S. TAKE THE OLD HARDCOPIES OF THE SCM01A AND SCM01B PRINTOUTS AND FILE THEM IN THE FOLDER MARKED SCM01A AND SCM01B HARDCOPY PRINTS.

****

ALL FILES FOR A NEW RELEASE ARE STORED ON DISK AND ARCHIVED ON A MAGNETIC TAPE VOLUME NAMED SCM001. THIS FIRST STEP WILL BE REFERRED TO AS INCREMENTAL BACKUP BECAUSE NEW RELEASES ARE WRITTEN TO VOLUME SCM001 ONE AT A TIME. AFTER SEVERAL RELEASES HAVE ACCUMULATED ON THE DISK, AND THEIR SOFTWARE IS STABLE SO IT IS NOT FREQUENTLY REVISED, THEY ARE COPIED TO ARCHIVE VOLUMES SCM01A AND SCM01B. THESE VOLUMES ARE MAINTAINED PERMANENTLY AND ARE NEVER ERASED. TAPE VOLUMES SCM001 AND SCM01B ARE LOCATED IN A LOCKED TAPE CABINET IN THE VAX COMPUTER ROOM. TAPE VOLUME SCM01A IS LOCATED IN THE SAFE. WITH THIS SCHEME, PRODUCT SOFTWARE IS ALWAYS BACKED UP IN TWO LOCATIONS, EITHER DISK AND SCM001 OR SCM01A AND SCM01B. AS MORE TAPE SPACE IS NEEDED, NEWER TAPE VOLUMES CAN BE NAMED SCM02A, SCM02B, ETC.

****

THE FIRST PROCEDURE IS THE ARCHIVING OF ALL FILES IN THE LOAD FILE DIRECTORY AFTER THE SOFTWARE HAS PASSED ACCEPTANCE TESTS AND THE MASTER CHIPS HAVE BEEN CREATED.

8888888888888888888888888888888888888888888888888888888888888888888888888888888888
THIS PROCEDURE IS USED AFTER THE LOAD FILES (720) HAVE BEEN RELEASED TO BLUEPRINT FOR DISTRIBUTION AND THERE ARE MANY 720'S IN THE [SOFTLIB] DIRECTORY. THESE 720 DIRECTORIES AND THEIR FILES WILL BE ARCHIVED ON TWO TAPES; SCM01A AND SCM01B. SCM01A IS HELD IN THE SAFE WHILE SCM01B IS HELD IN THE COMPUTER ROOM.

A. BEFORE YOU BEGIN COPYING FILES TO THE MAGNETIC TAPES YOU MUST 'ALLOCATE' THE TAPE DRIVE TO YOURSELF AND THE TERMINAL ON WHICH YOU ARE WORKING. THIS IS TO PREVENT ANYONE ELSE FROM OVERWRITING YOUR PARTICULAR TAPE WHILE YOU ARE USING IT. (IF THEY DID, ALL INFORMATION ON THAT TAPE WOULD BE PERMANENTLY LOST.) IN ORDER TO 'ALLOCATE' THE TAPE DRIVE (AFTER LOGGING ON) TYPE

```
$ ALLOCATE MTA0:
```

MTA0 STANDS FOR 'MAGNETIC TAPE NUMBER A0' AND MEANS THAT YOU ARE DESIGNATING A SPECIFIC TAPE DRIVE FOR YOUR USE ONLY.

B. LEAVE A NOTE ON THE TERMINAL THAT SAYS YOU WILL BE RIGHT BACK FROM CHECKING ON THE TAPE DRIVE. THIS IS SO SOMEONE DOESN'T LOG YOU OFF, THEREBYE DEALLOCATING THE TAPE DRIVE AND CREATING THE POSSIBILITY OF SOMEONE ELSE USIN(MTA0 AND ERASING YOUR TAPE.

C. GO TO THE COMPUTER ROOM AND GET THE SOFTWARE CONTROL BACKUP TAPE MARKED SCM01B FROM THE TAPE CABINET. (IF THE CABINET IS LOCKED HAVE ONE OF THI COMPUTER PERSONNEL UNLOCK IT FOR YOU.) THE TAPE IS ON A SMALL REEL.

D. LOCATE THE TAPE DRIVE CONNECTED TO SILVER. MAKE SURE THAT THE 'BOT' AND 'ON LINE' LIGHTS ARE OFF. OPEN THE PLASTIC RING THAT ENCIRCLES THE REEL AND REMOVE THE STATIC TAPE FROM THE TAPE. LOCATE AND INSERT THE 'WRITE RING' INTO THE BACK SIDE OF THE TAPE REEL. (IF THIS IS NOT DONE, THE COMPUTER WILL NOT BE ABLE TO WRITE ANYTHING ONTO THE TAPE.) OPEN THE TAPE DRIVE DOOR AND LOAD THE REEL WITH THE MAGNETIC TAPE COMING OVER THE REEL FROM THE LEFT. LOC THE REEL IN PLACE BY PRESSING IN THE RAISED SIDE OF THE BLACK PLASTIC LOCK IN THE CENTER OF THE THE TAPE HUB. INSERT ABOUT THREE TO FIVE INCHES OF TAPE INTO THE FEED MECHANISM.

E. CLOSE THE TAPE DRIVE DOOR. PRESS THE LOAD/REW BUTTON TO LOAD THE TAPE. WHEN THE TAPE HAS FINISHED LOADING AND THE 'BOT' LINE HAS COME ON PRES THE 'ON LINE' BUTTON. CHECK THAT BOTH THE 'BOT' AND 'ON LINE' LIGHTS ARE ON. THE TAPE DRIVE IS ON AND READY.

F. RETURN TO YOUR TERMINAL. TYPE

```
$MOUNT/FOREIGN MTA0:
```

THIS LOADS OR MOUNTS THE APPROPRIATE SOFTWARE (LOGICALLY AND PHYSICALLY) WITI THE APPROPRIATE FORMAT TO ALLOW YOU TO TRANSFER FILES FROM VAX TO THE END OF THE MAGNETIC TAPE (AREA JUST AFTER THE LAST ENTRY).

G. CONNECT YOUR TERMINAL TO THE PRINTER AND PRINT A COPY OF THE [SOFTLI DIRECTORY. GO THROUGH EACH 720 DIRECTORY AND PURGE THE FILES OF ALL UNNECESSARY MATERIALS.

H. EDIT THE SCM01B.COM FILE IN THE [SOFTLIB] DIRECTORY TO REFLECT ALL T 720 DIRECTORIES THAT YOU WANT TO PUT ON THE SCM01B TAPE. AS YOU INSERT THE NEW NUMBERS INTO THE SCM01B.COM FILE REMOVE THOSE LISTED THAT HAVE ALREADY BEEN PUT ON THE TAPE IN THE PREVIOUS SESSION.

I. WHEN YOUR EDITING SESSION IS COMPLETE AND THE SCM01B.COM CONTAINS ALL THE 720 NUMBERS THAT YOU WANT ARCHIVED TYPE

@SCM01B.COM

AT THE SAME TIME PRESS CTRL/PRINT TO RECEIVE A PRINTOUT OF THE NEWLY ARCHIVED DIRECTORIES AND FILES.

J. WHEN THE $ PROMPT RETURNS TYPE

$DISMOUNT/NOUNLOAD MTA0:

THIS WILL REWIND THE TAPE WITHOUT TAKING IT OFF-LINE OR UNLOADING IT. IT IS SIMILAR TO A REWIND COMMAND BUT THE USER MUST ISSUE A MOUNT COMMAND BEFORE USING THE TAPE AGAIN.

K. TYPE

$ MOUNT MTA0: SCM01B

THIS MOUNTS THE TAPE VOLUME ON DRIVE MTA0.

L. TYPE

$ DIR MTA0:

THIS WILL LIST ON HARD COPY ALL THE 720 DIRECTORIES THAT ARE NOW CONTAINED ON THE SCM01B TAPE.

M. TYPE

$TIME

THIS WILL GIVE THE EXACT TIME AND DATE ON THE HARDCOPY TO DETERMINE EXACTLY WHEN THE DIRECTORIES ON THIS TAPE WERE ARCHIVED.

N. TYPE

$SHOW PROC

THIS WILL SHOW THE DIRECTORIES USED, THE DEVICES ALLOCATED, AND WILL PROVIDE OTHER INFORMATION ON HARDCOPY TO BE USED IN CASE THE FILE EVER HAS TO BE THOROUGHLY LOOKED AT FOR SOME TYPE OF ERROR OR PROBLEM.

O. TYPE

$ DISMOUNT MTA0:

THIS COMMAND WILL REWIND THE TAPE, TAKE IT OFF-LINE, AND WILL UNLOAD IT FROM THE TAPE DRIVE.

P. GO TO THE SAFE AND LOCATE THE TAPE MARKED SCM01A. LEAVE A NOTE ON YOUR TERMINAL INDICATING WHERE YOU ARE AND GO BACK TO THE COMPUTER ROOM. REMOVE SCM01B FROM THE TAPE DRIVE, REMOVE THE WRITE RING, INSTALL THE WRITE RING ON SCM01A, MOUNT SCM01A THE SAME WAY THAT YOU MOUNTED SCM01B, AND PUT SCM01B AND THE NEW HARDCOPY OF THE .COM FILE BACK INTO THE CABINET. GO BACK TO YOUR TERMINAL AND REPEAT STEPS "D." THROUGH "O." SUBSTITUTING SCM01A WHERE SCM01B APPEARS.

Q. GO TO THE COMPUTER ROOM AFTER LEAVING A NOTE ON YOUR TERMINAL EXPLAINING WHERE YOU ARE. REMOVE SCM01A FROM THE TAPE DRIVE, REMOVE THE WRITE RING, REPLACE THE STATIC TAPE ON THE END OF THE MAGNETIC TAPE, REINSTALL THE OUTER PLASTIC RING ON THE TAPE REEL, AND RETURN THE TAPE AND THE NEW HARDCOPY PRINTOUT TO THE SAFE.

R. RETURN TO YOUR TERMINAL AND TYPE

```
$ DEALLOCATE MTA0:
```

THIS WILL DEALLOCATE THE TAPE DRIVE MTA0 SO THAT IT MAY BE USED BY OTHERS.

S. TAKE THE OLD HARDCOPIES OF THE SCM01A AND SCM01B PRINTOUTS AND FILE THEM IN THE FOLDER MARKED SCM01A AND SCM01B HARDCOPY PRINTS.

THIS CONCLUDES YOUR ARCHIVING PROCEDURES FOR THE 720 LOAD FILES.

NORMAL PROCEDURE FOR PROGRAMMING & VERIFYING FOUR MASTER UPROCESSORS USING THE KING PROM PROGRAMMER.

INTRODUCTION

THIS DOCUMENT IS INTENDED TO GIVE THE INEXPERIENCED USER A STEP-BY-STEP GUIDELINE FOR PROGRAMMING FOUR MASTER EPROMS AS PART OF A LOAD FILE RELEASE. IT IS ASSUMED THAT THE USER IS FAMILIAR WITH RUDIMENTARY PRINCIPLES OF DEALING WITH VAX. IT WILL BE EASIEST TO USE THIS PROCEDURE FOR SINGLE-.DOC-FILE, SINGLE-CHIP RELEASES; HOWEVER, AN EFFORT HAS BEEN MADE TO TAKE SOME POSSIBLE VARIATIONS INTO ACCOUNT, AND THESE ARE DEALT WITH IN MORE DETAIL IN SECTION 2.

EXAMPLES ARE PRESENTED AS FOLLOWS: THE ENTIRE LINE WHICH APPEARS ON THE SCREEN IS SHOWN HERE. IF PART OF THAT INFORMATION IS GENERATED WITHOUT ANY PROMPTING FROM THE USER, THE USER SIMPLY DOES NOT TYPE THAT PART OF THE LINE. FURTHER, WHEN PART OF A LINE APPEARS IN PARENTHESES AND IS PRECEDED BY AN ASTERISK, THAT INFORMATION VARIES AND IS GIVEN HERE AS AN EXAMPLE: THE USER IS RESPONSIBLE FOR OBTAINING AND TYPING THE CORRECT INFORMATION (HE DOES NOT TYPE THE ASTERISK).

IT IS ALSO ASSUMED THAT THE USER WILL PRESS THE 'RETURN' KEY AFTER EACH ENTRY.

IF YOU QUICKLY RUN INTO MAJOR OPERATIONAL PROBLEMS, CHECK THAT THE TERMINAL IS IN UPPER CASE (CAPS LOCK KEY DOWN), AT 2400 BAUD, AND SET TO JUMP SCROLL.

EPROMS ARE SENSITIVE TO STATIC ELECTRICITY; AVOID PLACING PROGRAMMABLE DEVICES ON SUCH STATIC CARRIERS AS PLASTIC AND PAPER. USE AN ANTI-STATIC MATERIAL (BLACK FOAM MAY BE OBTAINED FROM THE TOOL CRIB) AND STICK THE DEVICES PINS INTO THE MATERIAL. THIS WILL BOTH PROTECT THE PINS AND HELP KEEP THE DEVICE FROM FALLING ON THE FLOOR.

SECTION 1: BASIC PROCEDURE FOR PROGRAMMING MASTER CHIPS

YOU WILL NEED:

A) YOUR LOAD FILE RELEASE ORDER
B) APPROPRIATE BLANK EPROMS (4)
C) ANTI-STATIC FOAM
D) SELF-ADHESIVE LABELS
E) A PEN
F) CLEAR ADHESIVE TAPE

PROCEED:

1. LOG ON IN THE NORMAL MANNER AT A TERMINAL CONNECTED TO A KING PROM PROGRAMMER.

2. TYPE $SET TERM/NOBROADCAST
THIS PREVENTS ANY MESSAGES SENT TO YOUR USERNAME WHILE YOU ARE PROGRAMMING FROM BEING PROGRAMMED INTO THE CHIPS (EPROMS OR PROGRAMMABLE INTEGRATED CIRCUIT DEVICES).

3. BE SURE YOU ARE AWARE OF THE STARTING ADDRESS AND BUFFER LENGTH INFORMATION IN THE .COM FILE PERTAINING TO THE LOAD FILE RELEASE YOU ARE WORKING WITH.

4. TURN THE PROGRAMMER MAIN POWER SWITCH ON. SET THE PROGRAMMER PROG/VAX SWITCH TO PROG. PRESS THE PROGRAMMER RESET BUTTON AND WATCH THE TERMINAL SCREEN; THIS PROMPT SHOULD APPEAR:

CMD >

5. IF YOU ARE PROGRAMMING AN 8748, 8741, 8749, OR 8748H TYPE CHIP, BE SURE THAT THE PROGRAMMER'S SECONDARY POWER SWITCH IS OFF.

* NEVER PLUG ONE OF THE ABOVE CHIP TYPES INTO THE PROGRAMMER UNLESS THE SECONDARY POWER SWITCH IS OFF. SERIOUS DAMAGE COULD RESULT. *

IF YOU ARE USING ONE OF THE ABOVE CHIP TYPES, ALSO MAKE SURE THAT THE MICROPROCESSOR TYPE SWITCH IS POINTING TO THE CORRECT TYPE OF DEVICE. (UP FOR 8048 OR 8741; DOWN FOR 8749 OR 8048H.)

6. TYPE CMD > HS
THIS STANDS FOR 'HELP_ SET', AND IT MEANS THAT YOU NEED HELP WITH TH COMMAND 'SET'. THE TERMINAL WILL DISPLAY A LISTING OF THE VARIOUS TYPES OF PROGRAMMABLE CHIPS AND THEIR RESPECTIVE CODES. FOR INSTANCE THE CODE FOR AN 8048 CHIP IS C. NOTICE THE CODE FOR THE CHIP TYPE YOU ARE USING.

7. TYPE CMD > S
THIS STANDS FOR 'SET', AND IT MEANS THAT YOU WANT TO TELL THE PROGRAMMER WHAT TYPE OF CHIP IT WILL BE WORKING WITH AND TO 'SET' ITS CIRCUITS ACCORDINGLY.

8. TYPE EPROM TYPE * (C)
HERE YOU TYPE THE CODE FOR THE TYPE OF CHIP YOU ARE USING: C FOR AN 8048. USE THE CODE THAT YOU NOTED IN STEP 6. IF YOU HAVE FORGOTTEN, TYPE CMD > HS AGAIN & START OVER WITH STEP 6.

9. FIND THE SOCKET ON THE PROGRAMMER THAT IS MARKED FOR YOUR TYPE CHIP. MAKE SURE THAT THE SOCKET'S LEVER IS UP. (THIS UNLOCKS THE SOCKET.) ('UP' IN THIS CASE IS 'PERPENDICULAR TO THE SURFACE OF THE PROGRAMMER'.)

10. NOTICE THAT EACH OF YOUR FOUR MICROPROCESSORS HAS A LITTLE NOTCH ON ONE END. (IT'S A MARKER TO HELP LOCATE PIN #1 .) PLACE ONE CHIP INTO THE APPROPRIATE SOCKET WITH THE NOTCH ON THE HIGHER END OF THE CHIP. DO NOT FORCE THE CHIP INTO THE SOCKET. IT SHOULD DROP INTO PLACE EASILY. IF IT DOES NOT, CHECK THAT THE SOCKET LEVER IS UP AND TRY AGAIN.

11. MOVE THE SOCKET LEVER DOWN. THIS LOCKS THE SOCKET AND CONNECTS THE CHIP INTO THE PROGRAMMER'S CIRCUITS.

12. IF YOU ARE USING ONE OF THE CHIP TYPES MENTIONED IN STEP 5, TURN THE PROGRAMMER'S SECONDARY POWER SWITCH ON NOW.

13. TYPE CMD > B
THIS STANDS FOR 'BYTE CHECK' AND IT MEANS THAT YOU WANT THE PROGRAMMER TO CHECK THAT THE EPROM HAS BEEN FULLY ERASED BEFORE YOU TRY TO PROGRAM ANYTHING ELSE INTO IT. IT DOES THIS BY CHECKING THAT EVERY ADDRESS (LOCATION) IN THE CHIP HAS THE SAME THING IN IT.

14. TYPE EPROM ADDRESS: * (0)
HERE YOU TYPE THE NAME OF THE ADDRESS IN THE CHIP WHERE YOU WANT TH PROGRAMMER TO START CHECKING. NORMALLY, THIS WILL BE AT THE BEGINNING, NAMELY ADDRESS 0.

15. TYPE BUFFER LENGTH: * (400)
THE PROGRAMMER NEEDS TO KNOW HOW MANY LOCATIONS TO CHECK. YOU LET IT KNOW BY TYPING IN THE CHIP'S MEMORY SIZE. THIS WILL NORMALLY BE THE BUFFER LENGTH IN YOUR .COM FILE WHICH YOU NOTED IN STEP 3. OFTEN, IT WILL BE 400 OR 800 HEX.

16. TYPE VERIFY PATTERN: * (FF)
SOME CHIPS, WHEN FULLY ERASED, HAVE '00' AT EVERY ADDRESS. OTHER TYPES HAVE 'FF'. SEE TABLE 1 FOR A LIST OF WHICH PROGRAMMABLE DEVICES ERASE TO WHICH PATTERN. BY TYPING THAT INFORMATION HERE, YOU TELL THE PROGRAMMER WHAT TO CHECK FOR AT EACH ADDRESS.

17. STARTED-FINISHED
THIS IS WHAT THE TERMINAL SCREEN WILL DISPLAY IF EVERY ADDRESS IN THE CHIP HAS THE SPECIFIED PATTERN IN IT: THE CHIP IS COMPLETELY ERASED.
IF THE CHIP IS NOT COMPLETELY ERASED, THE TERMINAL SCREEN WILL DISPLAY STARTED- MISS AT LOC XXXX. THIS MEANS THAT AT LOCATION XXXX, THE PROGRAMMER FOUND SOME MATERIAL THAT DID NOT MATCH THE VERIFY PATTERN; THE CHIP IS THEREFORE NOT COMPLETELY ERASED.
IF THIS HAPPENS, FIRST CHECK THAT ALL POWER SWITCHES ARE ON, THAT ANY CHIP-TYPE SWITCHES ARE IN THE RIGHT POSITIONS, AND THAT THE SOCKET LEVER IS DOWN; THEN TRY STEPS 13 THROUGH 17 AGAIN. IF THE CHIP STILL FAILS, PLACE IT UNDER AN ULTRAVIOLET LIGHT FOR 45 MINUTES (MAKING SURE THE WINDOW IN THE TOP OF THE CHIP IS UNCOVERED). THEN TRY AGAIN. WHEN THE STARTED-FINISHED MESSAGE APPEARS, TURN THE SECONDARY POWER SWITCH OFF (IF APPLICABLE), UNLOCK THE SOCKET BY MOVING THE LEVER UP, REMOVE THE CHIP, AND PLACE IT ON ANTI-STATIC MATERIAL.

18. REPEAT STEPS 10 THROUGH 17 WITH EACH OF THE OTHER THREE CHIPS, MAKING SURE THAT THE SECONDARY POWER SWITCH (IF APPLICABLE) IS OFF BEFORE YOU REMOVE ANY CHIP FROM THE SOCKET, AND ON BEFORE YOU ATTEMPT A BYTE CHECK. ALSO REMEMBER TO LOCK AND UNLOCK THE SOCKET AS APPROPRIATE WITH THE SOCKET LEVER.

19. TYPE CMD > R
THIS STANDS FOR 'READ', AND IT MEANS THAT YOU ARE GOING TO PUT THE INFORMATION YOU WANT TO GO INTO THE CHIPS INTO THE PROGRAMMER, OR 'READ' THE INFORMATION FROM YOUR FILES ON VAX TO THE PROGRAMMER.

20. TYPE FILE ADDRESS: * (0)
HERE YOU TYPE THE ADDRESS IN YOUR VAX FILE WHERE YOU WANT THE PROGRAMMER TO BEGIN READING INFORMATION INTO ITS OWN MEMORY. NORMALLY FOR A SINGLE-CHIP DEVICE WITH A MEMORY OF LESS THAN 1000HEX, THIS WILL BE 0.

21. TYPE BUFFER LENGTH: * (400)
HERE AGAIN YOU TYPE THE SIZE OF THE INFORMATION THAT THE RAM NEEDS TO BE READY TO ACCEPT. THIS WILL AGAIN NORMALLY BE THE BUFFER SIZE IN YOUR .COM FILE THAT YOU NOTED IN STEP 3. HOWEVER, THERE MAY BE EXCEPTIONS, AS IN A CASE WHERE THE HEX FILE ON VAX IS TOO LONG FOR THE RAM'S MEMORY, AND YOU FIND IT NECESSARY TO READ THE INFORMATION INTO THE RAM IN STAGES.

22. TYPE * ($TYPE [SOFTLIB.720000101]722002401.HEX)
THIS TELLS THE PROGRAMMER WHICH OF YOUR FILES ON VAX IT IS SUPPOSED TO START LOADING INTO ITS MEMORY. NOTE THAT ALTHOUGH YOU ARE PROMPTED WITH THE DOLLAR SIGN ($), THE PROGRAMMER IS STILL IN CONTROL. YOU ACTUALLY TYPE THE WORD 'TYPE', FOLLOWED BY '[SOFTLIB. ETC', WHERE THE VARIABLES ARE THE 720- NUMBER AND THE 722- NUMBER. YOU CAN FIND BOTH OF THESE NUMBERS BY LOOKING ON THE LOAD FILE RELEASE ORDER.

AS YOUR VAX FILE IS COPIED INTO THE RAM, THE CURSOR ON THE TERMINAL WILL MOVE DOWN THE SCREEN; IT WILL LOOK AS THOUGH THE SREEN HAS GONE BLANK. WHEN THE COPYING IS COMPLETE, THE PROGRAMMER WILL PROMPT YOU AGAIN.

23. TYPE CMD > PV
THIS STANDS FOR 'PROGRAM- VERIFY' AND IT MEANS THAT YOU WANT THE PROGRAMMER TO PUT THE INFORMATION IN ITS MEMORY INTO THE CHIP, AND THEN TO GO BACK AND CHECK THAT IT HAS DONE SO CORRECTLY ACCORDING TO ITS INSTRUCTIONS.

24. TYPE EPROM ADDRESS: * (0)
HERE YOU TYPE THE ADDRESS IN THE CHIP WHERE YOU WANT THE PROGRAMMING TO BEGIN. NORMALLY THIS WILL BE 0, BUT SOMETIMES YOU MAY WANT TO START PRUGRAMMING IN THE MIDDLE OF THE CHIP- FOR INSTANCE, IN A CASE WHERE THE INFORMATION WAS TOO LONG TO FIT INTO THE PROGRAMMER ALL AT ONCE, SO YOU HAD TO PROGRAM YOUR CHIPS IN TWO STAGES.

25. TYPE BUFFER LENGTH: * (400)
HERE AGAIN YOU TYPE THE SIZE OF THE INFORMATION WHICH IS GOING TO BE READ INTO THE CHIP. THIS WILL AGAIN NORMALLY BE THE BUFFER SIZE IN YOUR .COM FILE THAT YOU NOTED IN STEP 3. HOWEVER, THERE MAY BE EXCEPTIONS, AS NOTED ABOVE.

26. TYPE RAM ADDRESS: * (0)
HERE YOU LET THE PROGRAMMER KNOW AT WHAT POINT IN ITS OWN MEMORY YOU WANT IT TO START UNLOADING THE INFORMATION. RAM STANDS FOR RANDOM ACCESS MEMORY, AND IT IS THE TYPE OF MEMORY CONTAINED IN THE PROGRAMMER. SOMETIMES YOU MIGHT HEAR THE TERMS 'PROGRAMMER' AND 'RAM' USED INTERCHANGEABLY. NORMALLY YOU WANT THE RAM TO BEGIN LOADING AT THE BEGINNING, NAMELY ADDRESS 0.

27. STARTED- FINISHED
VERIFY
STARTED- FINISHED
THE ABOVE MESSAGE WILL APPEAR AFTER A SUCCESSFUL PROGRAMMING SESSION.
IN THE EVENT OF A FAILURE, A 'MISS' MESSAGE WILL APPEAR INSTEAD OF THE 'FINISHED' MESSAGE. IF THIS HAPPENS BEFORE THE VERIFY MESSAGE APPEARS, THERE HAS BEEN A PROBLEM WITH THE PROGRAMMING. FIRST TURN OFF THE SECONDARY POWER SWITCH (IF APPLICABLE), UNLOCK THE SOCKET, REMOVE THE CHIP, AND REPLACE IT IN THE SOCKET, BEING SURE TO PUT THE NOTCH IN THE CHIP TOWARDS THE TOP OF THE PROGRAMMER. RELOCK THE SOCKET AND TURN THE SECONDARY POWER SWITCH ON IF APPLICABLE. THEN REPEAT THE BYTE CHECK PROCEDURE OF STEPS 13 THROUGH 17. IF THIS IS SUCCESSFUL (IF THE STARTED- FINISHED MESSAGE APPEARS), REPEAT THE 'CMD R' PROCEDURE OF STEP X, THEN REPEAT THE 'CMD > PV' PROCEDURE. IF THE BYTE CHECK IS NOT SUCCESSFUL, PLACE THE CHIP UNDER AN ULTRAVIOLET LIGHT FOR 45 MINUTES (MAKING SURE THE WINDOW IN THE TOP OF THE CHIP IS UNCOVERED). THEN REPEAT THIS PROCEDURE FROM THE BEGINNING.

28. TYPE CMD > F
THIS SAYS TO THE RAM, 'FILL YOUR MEMORY UP WITH THE PATTERN I'M ABOUT TO GIVE YOU'.

29. TYPE RAM ADDRESS: * (0)
THIS IS THE ADDRESS IN THE RAM WHERE YOU WANT THE FILLING PROCESS TO BEGIN.

30. TYPE BUFFER LENGTH: * (400)
HERE AGAIN YOU TYPE THE SIZE OF THE INFORMATION THAT THE RAM NEEDS TO BE READY TO ACCEPT. THIS WILL AGAIN NORMALLY BE THE BUFFER SIZE IN YOUR .COM FILE THAT YOU NOTED IN STEP 3.

31. TYPE FILL PATTERN: FF
THIS IS THE PATTERN WITH WHICH YOU WANT THE RAM TO FILL EVERY ADDRESS IN ITS MEMORY. THIS CMD > F ROUTINE HELPS IN POINTING UP ERRORS IN, FOR INSTANCE, BUFFER SIZES IN THE .COM FILE, WHICH MIGHT NOT OTHERWISE BECOME EVIDENT UNTIL QUITE A LOT OF TIME HAD ALREADY BEEN INVESTED IN PROGRAMMING CHIPS INCORRECTLY.

32. REPEAT THE CMD > R PROCEDURE OF STEPS 19 THROUGH 22. THIS IS AN INTERNAL DOUBLE-CHECK TO ENSURE THAT YOU DIDN'T MAKE A SILLY MISTAKE LIKE TYPING THE WRONG NUMBER WHEN YOU COPIED YOUR FILE INTO THE RAM IN STEP 22. BY REPEATING THE CMD> R PROCEDURE, YOU ARE EFFECTIVELY REFILLING THE CONTENTS OF THE RAM WITH THE SAME THING THAT WAS REMOVED WHEN YOU FILLED THE RAM WITH FFS. YOU ARE NOW GOING TO CHECK THAT THE CURRENT RAM CONTENTS ARE THE SAME AS THEY WERE BEFORE.

33. TYPE CMD > V
THIS IS 'VERIFY', AND IT TELLS THE PROGRAMMER THAT YOU WANT TO CHECK THAT THE INFORMATION IN THE DEVICE IT JUST PROGRAMMED IS IDENTICAL TO THE INFORMATION THAT IT JUST READ FROM YOUR FILE ON VAX. YOU WILL AGAIN BE PROMPTED FOR SIZE AND ADDRESS INFORMATION, AND AFTER A SUCCESSFUL COMPARISON THE STARTED- FINISHED MESSAGE WILL AGAIN APPEAR.
AT THIS POINT TURN THE SECONDARY POWER SWITCH OFF IF APPLICABLE, UNLOCK THE SOCKET, REMOVE THE CHIP AND PLACE IT ON ANTISTATIC MATERIAL. IF THE STARTED- FINISHED MESSAGE IS NOT FORTHCOMING, REPEAT THE CMD > R PROCEDURE AS ABOVE, THEN REPEAT THE CMD > V PROCEDURE. IF THE VERIFY CHECK STILL FAILS, ERASE THE EPROM AS IN STEP 27 AND START OVER AGAIN FROM THE BEGINNING.

34. REPEAT STEPS 23 THROUGH 27 WITH THE OTHER THREE MASTER CHIPS, MAKING SURE THAT THE SECONDARY POWER SWITCH IS TURNED ON OR OFF AND THAT THE SOCKET IS LOCKED OR UNLOCKED AS APPROPRIATE.

35. REPEAT THE CMD > F PROCEDURE OF STEPS 28 THROUGH 31, THEN THE CMD > R PROCEDURE OF STEPS 19 THROUGH 22.

36. REPEAT THE CMD > V PROCEDURE OF STEP 33 WITH EACH OF YOUR FOUR MASTER CHIPS, MAKING SURE THAT THE SECONDARY POWER SWITCH AND SOCKET LOCKING SWITCH ARE SET APPROPRIATELY.

37. SET THE PROGRAMMER PROG/VAX SWTICH TO VAX. THE DOLLAR SIGN ($) SHOULD APPEAR.

38. SET THE MAIN PROGRAMMER POWER SWITCH OFF.

39. PREPARE FOUR 1.75" X .5" SELF-ADHESIVE LABELS AS FOLLOWS:

122-XXXX-XX
MASTER
DAY MO YR

WHERE THE 122- NUMBER IS THE NUMBER ON THE LOAD FILE RELEASE ORDER ASSOCIATED WITH THE .HEX FILE WHICH YOU READ FROM VAX INTO THE PROGRAMMER; AND DAY/MO/YR IS THE DATE THE PROGRAMMING WAS COMPLETED. PLEASE USE LETTERS RATHER THAN NUMBERS TO DESIGNATE THE DATE (EG: USE 16 MARCH 1958, NOT 16-3-58).

40. PLACE THE LABEL ON THE CHIP WITH THE CHIP NOTCH TO THE LEFT AS YOU READ THE LABEL.

41. COVER EACH LABEL WITH CLEAR ADHESIVE TAPE, BEING CAREFUL NOT TO TOUCH THE INTEGRATED CIRCUIT'S PINS UNNECESSARILY. (COMMON TAPE DISPENSERS AND TAPE ARE GOOD GENERATORS OF STATIC ELECTRICITY; STATIC ELECTRICITY CAN DAMAGE PROGRAMMED DEVICES OR CHANGE THEIR CONTENTS.)

****
YOU HAVE NOW COMPLETED THE PROGRAMMING AND LABELING OF FOUR MASTER CHIPS. PROCEED FROM HERE AS OUTLINED IN THE LOAD FILE RELEASE ORDER HANDLING PROCEDURE.
****

+ + + + + + + + + + + + BASIC PROMPTS + + + + + + + + + + + +

ALWAYS REMEMBER THE MEANING OF THE BASIC PROMPTS WHICH THE PROGRAMMER WILL GIVE YOU:

FILE ADDRESS
THIS ALWAYS REFERS TO A LOCATION (AN ADDRESS) IN A FILE ON VAX. WHEN YOU ARE PROGRAMMING MASTER CHIPS, THE FILE WILL ALMOST CERTAINLY REFER TO A .HEX FI THE PROGRAMMER NEEDS TO KNOW WHERE IN THE .HEX FILE TO BEGIN READING INFORMATION INTO ITS OWN MEMORY.

RAM ADDRESS
THIS ALWAYS REFERS TO AN ADDRESS IN RAM, THAT IS IN THE PROGRAMMER. THE PROGRAMMER NEEDS TO KNOW WHERE IN ITS OWN MEMORY TO BEGIN LOADING OR UNLOADING INFORMATION.

EPROM ADDRESS
THIS ALWAYS REFERS TO AN ADDRESS IN THE CHIP. THE PROGRAMMER NEEDS TO KNOW WHERE IN THE CHIP TO BEGIN LOADING OR UNLOADING INFORMATION.

BUFFER LENGTH
THIS ALWAYS REFERS TO THE SIZE OF THE MATERIAL THAT IS TO BE DEALT WITH: 'HOW LONG IS IT?'. REMEMBER ALWAYS TO GIVE THE ANSWER IN HEXADECIMAL NOTATION.

+ + + + + + + + + + + COMMANDS SYNOPSIS + + + + + + + + + + +

CMD >	B	'BYTE CHECK'.		CHECKS THAT ALL EPROM ADDRESSES CONTAIN SPECIFIED PATTERN.
CMD >	C	'COPY'	.	COPY CONTENTS OF EPROM TO RAM BUFFER.
CMD >	D	'DISPLAY'	.	TYPE CONTENTS OF RAM BUFFER ON TERMINAL SCREEN.
CMD >	E	'EPROM'	.	TYPE CONTENTS OF EPROM ON TERMINAL SCREEN.
CMD >	F	'FILL'	.	FILL THE RAM BUFFER WITH THE PATTERN TO BE SPECIFI
CMD >	M	'MOVE'	.	MOVE DATA FROM ONE AREA OF RAM BUFFER TO ANOTHER.
CMD >	P	'PROGRAM'	.	FILL THE EPROM WITH A COPY OF THE RAM CONTENTS, WITHIN THE ADDRESS LIMITS SPECIFIED.
CMD >	R	'READ'	.	READ (COPY) A TO-BE-SPECIFIED FILE FROM VAX TO THE RAM BUFFER.
CMD >	S	'SET'	.	SET THE PROGRAMMER FOR THE TYPE OF DEVICE TO BE PROGRAMMED.
CMD >	T	'TRANSMIT'	.	ALLOWS AN OBJECT FILE TO BE TRANSFERRED FROM RAM TO VAX.
CMD >	V	'VERIFY'	.	VERIFY THAT EPROM CONTENTS MATCH RAM CONTENTS, WITHIN THE ADDRESS LIMITS SPECIFIED.

CMD > XXX/ 'OPEN X' . OPEN LOCATION XXX, THAT IS PLACE IN A POSITION TO MODIFY ADDRESS XXX.

CMD > CTRLX . CAUSES PROGRAMMER TO ENTER MODE ALLOWING DIRECT USER-TO-VAX COMMUNICATION WHILE IN PROGRAMMER MODE.

+ + + + + + + + + + + + + VARIATIONS + + + + + + + + + + + + +

SUPPOSE YOU NEED TO PROGRAM A CHIP, AND THE HEX FILE IS 2K WHILE THE RAM MEMORY IS ONLY 1K. OBVIOUSLY THE WHOLE FILE WILL NOT FIT INTO THE RAM AT ONCE. USE WHAT YOU KNOW ABOUT THE BASIC COMMANDS TO FILL THE RAM AND PROGRAM THE CHIP IN STAGES:

1. READ YOUR FILE FROM VAX TO THE RAM. TELL IT TO START AT ADDRESS 0 AND TO EXPECT A FILE THAT IS 1K LONG. (REMEMBER THAT THE FILE IS REALLY 2K.) THE PROGRAMMER WILL READ THE FIRST 1K FROM YOUR .HEX FILE TO THE RAM.

2. WHEN YOU PROGRAM, TELL THE PROGRAMMER TO BEGIN AT ADDRESS 0 IN THE RAM AND IN THE CHIP, AND TO EXPECT A FILE THAT IS 1K (HEX) LONG. THE PROGRAMMER WILL PROGRAM YOUR CHIP UP TO ADDRESS 1000 WITH THE INFORMATION IN RAM.

3. NOW READ THE OTHER HALF OF YOUR .HEX FILE INTO RAM. DO THIS BY TELLING THE PROGRAMMER TO START READING AT ADDRESS 1000 AND TO EXPECT A FILE THAT IS 1K (HEX) LONG. NOTICE THAT THIS WILL CAUSE THE SECOND HALF OF YOUR .HEX FILE TO BE READ INTO RAM, REPLACING THE INFORMATION THAT WAS THERE BEFORE.

4. NOW PROGRAM THE SECOND HALF OF YOUR CHIP. DO THIS BY SPECIFYING TO THE PROGRAMMER TO START AT ADDRESS 0 IN THE RAM; TO START LOADING AT ADDRESS 1000 IN THE CHIP (REMEMBER THAT THE FIRST 1000 ADDRESSES IN THE CHIP ARE ALREADY FILLED); AND TO EXPECT TO HANDLE A FILE THAT IS 1K (HEX) LONG.

NOTICE THAT BY DOING THIS YOU HAVE SUCCESSIVELY READ AND PROGRAMMED BOTH HALVES OF YOUR .HEX FILE, USING THE 1K RAM MEMORY AS AN INTERMEDIARY BETWEEN THE 2K MEMORIES OF YOUR .HEX FILE AND OF THE CHIP.

+ + + + + + + + + + + + + + + THE END + + + + + + + + + + + + + + +

****
PROCEDURE FOR SOFTWARE CONTROL TO SUPERVISE A LOAD FILE RELEASE
****

THIS DOCUMENT IS INTENDED TO PROVIDE INSTRUCTIONS FOR THE SOFTWARE CONTROL REPRESENTATIVE (HEREINAFTER REFERRED TO AS 'YOU') TO SUPERVISE A LOAD FILE RELEASE FROM THE TIME HE RECEIVES THE LOAD FILE RELEASE ORDER TO THE TIME THE RELEASE IS COMPLETED. IT ATTEMPTS TO DEAL PRIMARILY WITH THE MECHANICS OF SUPERVISION; FOR EXPLANATIONS OF THE THEORIES AND PHILOSOPHICAL CONCEPTS BEHIND THE ACTIVITIES DESCRIBED HEREIN, SEE THE APPROPRIATE SOFTWARE CONTROL DOCUMENTS.

THIS DOCUMENT ASSUMES THAT YOU WILL BE USING THE FORM ENTITLED 'LOAD FILE RELEASES IN PROCESS' TO MONITOR YOUR PROGRESS. THE HEADINGS IN THIS DOCUMENT THEREFORE CORRESPOND TO THE HEADINGS ON THE FORM. IT IS ASSUMED THAT AS YOU COMPLETE A STEP, YOU WILL FILL IN THE DATE YOU COMPLETED IT IN THE SPACE BELOW THE APPROPRIATE COLUMN HEADING.

1. DATE RECEIVED
UPON RECEIVING A LOAD FILE RELEASE ORDER (FROM THE SOFTWARE PROJECT ENGINEER FOR NEW RELEASES; FROM THE ECO REPRESENTATIVE FOR REVISIONS TO EXISTING SOFTWARE), LOG IN THE LOAD FILE NUMBER AND UNIT NAME IN THE LEFTMOST SECTION OF THE FORM LABELED 'LOAD FILE #'. ALSO LOG IN THE DATE YOU RECEIVED THE RELEASE ORDER IN THE FIRST COLUMN, LABELED 'DATE RECEIVED'. IF THE LOAD FILE RELEASE IS FOR A REVISION TO EXISTING SOFTWARE, MAKE TWO COPIES OF THE DOCUMENT AND TAKE ONE TO JAN ODELL (ENGINEERING OFFICE) AND ONE TO PJ (COMPONENT ENGINEERING); THEY WILL USE THE INFORMATION ON THE RELEASE ORDER TO LOG THE NEW PART NUMBERS ON VAX AND TO GENERATE THE APPROPRIATE 122- PIECE PARTS PRINT.

2. ACCURACY & COMPLETENESS CHECK
CHECK THE RELEASE ORDER FOR COMPLETENESS AND ACCURACY. THE EASIEST WAY TO DO THIS MAY BE TO REFER TO THE DOCUMENT THAT THE SOFTWARE PROJECT ENGINEER SHOULD HAVE USED TO FILL IN THE RELEASE ORDER: 'SOFTWARE RELEASE PROCEDURE FOR BEGINNERS'. BE CONSTANTLY ALERT FOR ITEMS WHICH MAY SUPERFICIALLY LOOK RIGHT OR AT LEAST POSSIBLE, BUT WHICH ARE REALLY THE RESULT OF A MISUNDERSTANDING ON THE ENGINEER'S PART. (SUCH AS THE IC TYPE 8051 FOR A SUPPOSEDLY ELECTRICALLY PROGRAMMED PART, OR A 720- NUMBER IN PLACE OF A 722- NUMBER.)

3. CLEAR UP ANY DIFFICULTIES, INCONSISTENCIES OR OMISSIONS WITH THE SOFTWAR PROJECT ENGINEER. TAKE ADVANTAGE OF THIS TIME TO ASK HIM IF HE HAS TRANSFERRED HIS SOURCE AND COMMAND FILES TO SILVER::[EXCHANGE] FOR YOU. (IN PRACTICE AT THE DATE OF WRITING (3-11-83) SOFTWARE CONTROL HAS BEEN GENERATIN COMMAND FILES. THE ULTIMATE GOAL IS FOR THE ENGINEER TO DESIGN AND RUN THE COMMAND FILE, THEN USE THE RESULTING CODE IN ENGINEERING UNITS, SO THAT THERE CAN BE NO DOUBT THAT THE CODE CREATED BY THE COMMAND FILE IS VALID.)

4. CREATE SUB-DIR
CREATE A SUB-DIRECTORY IN [SOFTLIB] TO USE WITH THIS LOAD FILE RELEASE. DO THIS, WHEN LOGGED ON VAX, BY TYPING CREATE/DIR [SOFTLIB.720LOADFILE#] , WHERE THE 720- NUMBER IN THE SUB-DIRECTORY NAME IS THE NUMBER ON THE LOAD FI RELEASE ORDER (OMIT THE DASHES IN THE 720- NUMBER WHEN CREATING THE DIRECTORY). IF YOU NOW COMMAND $DIR (ASSUMING YOU ARE IN [SOFTLIB]), THERE WILL BE A NEW SUBDIRECTORY LISTED AS, FOR INSTANCE, 720001401.DIR . IF YOU TYPE $DIR [.720001401], THE TERMINAL WILL DISPLAY THE 'NO FILES FOUND' MESSA WHICH IS LOGICAL SINCE ALTHOUGH YOU HAVE CREATED THE DIRECTORY, YOU HAVEN'T PUT ANYTHING INTO IT YET.

5. COPY SOURCE & .COM FROM EXCHANGE

TYPE $DIR [EXCHANGE] TO CHECK THAT THE SOFTWARE PROJECT ENGINEER TRANSFERRED HIS SOURCE AND COMMAND FILES TO [EXCHANGE] FOR YOU. (THE NAMES OF THESE FILES WILL BE ON THE LOAD FILE RELEASE ORDER.) ASSUMING THAT THEY ARE THERE, TYPE THE FOLLOWING SEQUENCE:

$COPY [EXCHANGE]SOURCEFILENAME.EXT [SOFTLIB.720001401]SOURCEFILENAME.EXT

-- TYPE THE SOURCE FILE NAME AS IT APPEARS IN THE [EXCHANGE] DIRECTORY AND ON THE LOAD FILE RELEASE ORDER; TYPE THE 720- NUMBER CORRESPONDING TO THE SUBDIRECTORY WHICH YOU CREATED IN STEP 3 ABOVE.

REPEAT THE COPYING PROCESS FOR THE COMMAND (.COM) FILE. (IN REAL LIFE AT THE DATE OF WRITING, IT IS USUALLY NECESSARY FOR SOFTWARE CONTROL TO CREATE THE .COM FILE. FOR HELP WITH THIS, SEE [SOFTLIB.DOC.PROCS]SUPTSOFT.DEB .)

6. REQUEST BLANK CHIP SET(S)

SOFTWARE CONTROL TRIES TO KEEP A SUPPLY OF COMMONLY-USED PROGRAMMABLE DEVICES ON HAND. HOWEVER, WHEN THESE CHIPS ARE USED THEY NEED TO BE REPLACED; AND IF THERE ARE NONE ON HAND IN THE FIRST PLACE, THEY NEED TO BE OBTAINED FROM SOMEWHERE.

STANDARD PROCEDURE IS TO CALL JIM SHIELDS IN PURCHASING (EXT. 2530) AND REQUEST FOUR CHIPS. GIVE HIM THE PART NUMBER ON THE LOAD FILE RELEASE ORDER. USUALLY, YOU WILL ASK HIM TO HAVE THE CHIPS DELIVERED TO THE ENGINEERING DROP (IN THE ENGINEERING LIBRARY); HOWEVER, IF YOU EVER NEED CHIPS RIGHT AWAY, YOU MAY ASK HIM TO HAVE THE CHIPS READY FOR YOU IN MAIN STOCK, AND GO DOWN YOURSELF TO PICK THEM UP.

7. CREATE .COM FILE

IN THE IDEAL WORLD, YOU WILL NOT BE CREATING A COMMAND FILE; YOU WILL SIMPLY BE CHECKING THE ENGINEER'S COMMAND (.COM) FILE TO SEE THAT IT MEETS OUR REQUIREMENTS FOR REFERENCES TO 720- NUMBERS, VERSIONS OF SUPPORT SOFTWARE USED, AND ASSIGNING OF THE NAME MICRO$DISK TO [SOFTLIB] FILES. SEE [SOFTLIB.DOC.PROCS]SUPTSOFT.DEB FOR A MORE COMPLETE EXPLANATION OF THE .COM FILE AND OF HOW IT SHOULD BE DESIGNED.

8. SOURCE HEADING; .COM ACCURACY

CHECK THE SOURCE FILE WHICH YOU HAVE COPIED FROM [EXCHANGE] TO SEE THAT THE ENGINEER HAS PUT A HEADING ON IT, AND THAT THE HEADING SUPPLIED MEETS SOFTWARE CONTROL REQUIREMENTS. SEE [SOFTLIB]HEADER.LIS FOR A SAMPLE HEADING. WORK WITH THE SOFTWARE PROJECT ENGINEER TO CORRECT THE SOURCE FILE HEADING AND TO RESOLVE ANY DIFFICULTIES YOU HAD WITH THE .COM FILE. DO NOT EDIT THE FILES YOURSELF TO CORRECT THEM; HAVE THE ENGINEER MAKE CORRECTIONS AND THEN RE-TRANSFER THE FILES TO [EXCHANGE] FOR YOU.

9. RUN .COM FILE

A STANDARD COMMAND FILE FOR A SINGLE-CHIP PROCESSOR SYSTEM IS A LIST OF COMMANDS WHICH WILL GENERATE AN .LOA FILE (THE RESULT OF THE ASSEMBLER); A .DOC FILE (THE RESULT OF THE PARTITIONER); AND A .HEX FILE (THE RESULT OF THE FORMATTER). MORE COMPLEX SYSTEMS WILL OF COURSE HAVE MORE COMPLEX COMMAND FILES. (SEE [SOFTLIB.DOC.PROCS]SUPTSOFT.DEB FOR A MORE COMPLETE DISCUSSION.) WHEN YOU 'RUN' THE COMMAND FILE YOU ARE IN EFFECT SAYING TO VAX, 'DO WHAT THE COMMAND FILE TELLS YOU TO DO'. SINCE THE COMMAND TELLS IT TO CREATE FILES, YOU WILL HAVE MORE FILES AFTER RUNNING THE COMMAND FILE THAN YOU HAD BEFORE.

RUN THE COMMAND FILE BY TYPING $@XXXXXX.COM, WHERE XXXXX.COM IS THE NA OF THE APPRORITE COMMAND FILE. NOTICE THAT '@' IS THE COMMAND TO RUN A COMMA FILE. VAX WILL GIVE YOU CERTAIN INFORMATION AS IT RUNS THE COMMAND FILE. IT WILL SAY 'PREVIOUS LOGICAL NAME ASSIGNMENT RELACED' WHEN IT REDEFINES 'MICRO$DISK' ; IT WILL SAY 'ASSEMBLY COMPLETE WITH 0 ERRORS' WHEN IT HAS FINISHED RUNNING THE ASSEMBLER; AND IT WILL PRINT THE PROMPTS FOR RUNNING TH PARTITIONER AND FORMATTER THAT YOU WOULD HAVE HAD TO TYPE IN IF THEY HADN'T ALREADY BEEN IN THE COMMAND FILE.

WHEN THE ENTIRE COMMAND FILE HAS BEEN RUN AND YOU ARE AGAIN PROMPTED WITH THE '$' PROMPT, TYPE $DIR AND CHECK THAT THE FILES THAT WERE SUPPOSED TO BE CREATED WERE IN FACT CREATED.

10. PUT .HEX IN EXCHANGE

THE ENGINEER MUST NOW CHECK THAT THE .HEX FILE YOU HAVE JUST CREATED IS IDENTICAL THE .HEX FILE HE USED TO PROGRAM ENGINEERING CHIPS. IN ORDER TO 'GIVE' THE FILE TO HIM, YOU COPY IT INTO [EXCHANGE]. IF THE ENGINEER'S ACCOUNT IS ON THE GOLD NODE OF VAX, COPY THE .HEX FILE AS FOLLOWS:

```
COPY [SOFTLIB.720XXXX0X]722XXXXX.HEX GOLD::ENG$USERDISK:[EXCHANGE]
```

THE 720- NUMBER WILL BE THE SUBDIRECTORY NUMBER; THE 722- NUMBER WILL BE THE .HEX FILE NUMBER.
IF THE ENGINEER'S ACCOUNT IS ON SILVER, PROCEED AS ABOVE BUT OMIT 'GOLD::ENG$USERDISK:' .

11. DESIGNER RUNS 'DIFFERENCES'

THE MECHANISM FOR COMPARING TWO FILES TO SEE IF THEY ARE IDENTICAL IS A VAX COMMAND CALLED 'DIFFERENCES'. IN THE IDEAL SITUATION, THE DESIGNER WILL RUN DIFFERENCES BETWEEN HIS .HEX FILE AND YOUR .HEX FILE, AND THE RESULT WILL BE 'NO DIFFERENCES FOUND'. IN REAL LIFE AT THE TIME OF WRITING, THE ENGINEER IS OFTEN COMPARING YOUR .HEX FILE TO HIS .LOA FILE, WHICH CAN CAUSE VAX TO SEE DIFFERENCES. WHEN THIS HAPPENS, THE ENGINEER MUST SATISFY HIMSELF THAT THE DIFFERENCES ARE DIFFERENCES OF FORM AND NOT OF SUBSTANCE. WHEN THE ENGINEER TELLS YOU THAT YOUR .HEX FILE IS ACCURATE, YOU MAY PROCEED WITH THE NEXT STEP.

12. OBTAIN BLANK CHIP SETS

BY THIS TIME YOU SHOULD ALREADY HAVE THE CHIPS YOU HAD REQUESTED FROM JIM SHIELDS OR A SET FROM THE SOFTWARE CONTROL SUPPLIES. IF YOU DO NOT, CALL JIM SHIELDS AGAIN; IF HE CANNOT HELP YOU, SEE IF THE SOFTWARE PROJECT ENGINEER CAN.

13. PROGRAM & VERIFY FOUR MASTERS

FOR EACH PROGRAMMBABLE DEVICE IN EVERY LOAD FILE RELEASE YOU SUPERVISE, YOU WILL CREATE 4 MASTER CHIPS. THESE WILL ULTIMATELY BE DELIVERED AS FOLLOWS: ONE TO THE SOFTWARE PROJECT ENGINEER; ONE TO COMPONENTS FOR PERMANENT STORAGE; AND TWO TO RECEIVING INSPECTION FOR PROGRAMMING AND VERIFICATION OF PRODUCTION CHIPS. THE ACCURACY OF THE MASTERS IS OBVIOUSLY OF PARAMOUNT IMPORTANCE. FOR A DETAILED PROCEDURE FOR PROGRAMMING THE MASTER CHIPS FOR A LOAD FILE RELEASE, SEE [SOFTLIB.DOC.PROCS]PV.DEB .

14. DESIGNER VERIFIES & VALIDATES

THIS MEANS THAT THE SOFTWARE PROJECT ENGINEER MUST CHECK THE MASTER CHIP BOTH ELECTRICALLY AND FUNCTIONALLY. HE IS EXPECTED TO CHECK THAT THE CODE IN THE CHIP IS THE DESIRED CODE; AND HE IS EXPECTED TO PLUG THE CHIP INTO A WORKING ENGINEERING UNIT AND TEST IT THOROUGHLY FOR PRECISE AND ACCURATE FUNCTIONING. WAIT UNTIL THE ENGINEER TELLS YOU THAT THE CHIP IS SATISFACTORY BEFORE YOU PROCEED. (THE ENGINEER MAY KEEP HIS MASTER CHIP FROM THIS TIME ON.)

15. ARCHIVE

THIS IS THE PROCESS WHEREBY ALL THE FILES WHICH HAVE BEEN CREATED FOR AND USED IN THE LOAD FILE RELEASE ARE COPIED ONTO MAGNETIC TAPE FOR SAFE AND PERMANENT STORAGE. ARCHIVING IS DONE IN TWO STAGES. DURING EACH LOAD FILE RELEASE, THERE IS AN INITIAL ARCHIVING PROCEDURE WHICH COPIES ALL THE RELEVAN FILES FROM VAX TO MAGNETIC TAPE. HOWEVER, THE RELEVANT FILES ARE NOT DELETED FROM VAX AT THIS TIME. WHEN A SUFFICIENT NUMBER (ASK STEVE HOW MANY) OF LOAD FILE RELEASES HAVE BEEN PROCESSED AND TEMPORARILY ARCHIVED, ALL THE RELEVANT FILES FOR EACH RELEASE WILL BE PERMANENTLY ARCHIVED ON ANOTHER TWO MAGNETIC TAPES. (THE DUPLICATION IS OF COURSE TO PREVENT THE UNLIKELY DESTRUCTION OF ONE TAPE FROM CAUSING THE PERMANENT LOSS OF THE FILES.)

IT MAY BE HELPFUL TO YOU TO HAVE READ [STEVE.GUIDE]TAPEGUIDE.LIS, A LISTING OF COMMANDS USED DURING THE TRANSFERRING OF FILES FROM VAX TO MAGNETI TAPE AND VICE VERSA. YOU SHOULD CERTAINLY HAVE IT NEARBY AS AN ADDITIONAL REFERENCE WHILE YOU DO YOUR ARCHIVING.

USE THE FOLLOWING PROCEDURE TO ARCHIVE A LOAD FILE RELEASE, THAT IS TO COPY THE APPROPRIATE FILES FROM VAX TO MAGNETIC TAPE.

A. BEFORE YOU BEGIN ACTUALLY COPYING FILES TO MAGNETIC TAPE, YOU NEED TO 'ALLOCATE' THE TAPE DRIVE (MACHINE THAT THE TAPE PLUGS INTO) TO YOURSELF AND YOUR TERMINAL SO THAT NO ONE ELSE CAN ACCIDENTALLY START USING IT WHILE YOU ARE COPYING FILES. (IF THEY DID, YOU COULD LOSE EVERYTHING ON YOUR TAPE.) TO DO THIS, WHEN LOGGED ON VAX, TYPE

$ ALLOCATE MTA0

'MTA0' STANDS FOR 'MAGNETIC TAPE NUMBER A0', AND IT IS THE SPECIFIC TAPE DRIVE THAT YOU WILL BE USING TO INTERFACE THE TAPE WITH YOUR FILES ON VAX.

B. LEAVE A NOTE ON YOUR TERMINAL SAYING THAT YOU WILL BE RIGHT BACK FROM CHECKING ON THE TAPE DRIVE (SO THAT A FRUSTRATED FELLOW VAX USER WILL NOT LOG YOU OFF, THUS DE-ALLOCATING MTA0 AND ALLOWING THE POSSIBILITY THAT ANOTHER USER COULD TRY TO USE MTA0 AND ERASE YOUR TAPE).

C. GO TO THE COMPUTER ROOM AND GET THE SOFTWARE CONTROL TEMPORARY ARCHIVE (A LARGE TAPE REEL CALLED SCM001) FROM THE CABINET. YOU MAY NEED TO ASK ONE OF THE COMPUTER PERSONNEL TO UNLOCK THE CABINET FOR YOU.

D. FIND THE TAPE DRIVE CONNECTED TO SILVER. CHECK THAT THE 'BOT' AND 'ON LINE' LIGHTS ARE OFF. OPEN THE PLASTIC CASE THAT SCM001 IS IN BY TURNING THE PLASTIC LOCK IN THE CENTER OF THE CASE. REMOVE THE STATIC TAPE AND ANY STYROFOAM FROM THE END OF THE MAGNETIC TAPE ON THE SCM001 REEL. OPEN THE TAPE DRIVE AND LOAD THE REEL WITH THE MAGNETIC TAPE COMING OVER THE TOP OF THE REEL TO THE RIGHT. LOCK THE REEL IN PLACE BY PRESSING DOWN THE RAISED SIDE OF THE BLACK PLASTIC LOCK IN THE CENTER OF THE TAPE MOUNT. IF YOU ARE WORKING WITH A SMALL REEL, THREAD THE MAGNETIC TAPE THROUGH THE THREADING MECHANISM UNTIL APPROXIMATELY FOUR INCHES OF TAPE ARE SHOWING BELOW THE MECHANISM. (THE LARGE REEL THREADS AUTOMATICALLY.)

E. CLOSE THE TAPE DRIVE. PRESS THE LOAD/REW BUTTON. (THIS LOADS THE TAPE.) WAIT UNTIL THE TAPE HAS FINISHED LOADING AND THE 'BOT' LIGHT HAS COME ON. PRESS THE ON LINE BUTTON. THE TAPE DRIVE IS NOW ON AND READY. BOTH THE 'BOT' AND 'ON LINE' LIGHTS SHOULD NOW BE ON.

F. RETURN TO YOUR VAX TERMINAL. TYPE $MOUNT/FOREIGN MTA0: SCM001. THIS MOUNTS OR LOADS THE APPROPRIATE SOFTWARE (OR CONNECTS THE TAPE 'LOGICALLY' AS WELL AS PHYSICALLY) WITH THE APPROPRIATE FORMAT TO ALLOW YOU TO TRANSFER FILES FROM VAX TO THE END OF THE MAGNETIC TAPE.

G. TYPE $BACKUP/NOREWIND/LOG DRA1:[SOFTLIB.720LDFILE#] MTA0:720LDFILE#.BCK

THE WORD 'BACKUP' HERE REFERS TO A SECONDARY SUPPLY, SOURCE OR COPY, AND NOT TO THE ACTION OF MOVING BACKWARDS. THIS COMMAND STRING IS SIMILAR TO A 'COPY' COMMAND ON VAX. THE /NOREWIND/ QUALIFIER PREVENTS THE INFORMATION FROM BEING COPIED OVER ALREADY-EXISTING FILES. THE /LOG/ QUALIFIER IS A REQUEST THAT THE INFORMATION AS TO WHAT FILES ARE BEING COPIED BE DISPLAYED ON THE TERMINAL.
THE 'DRA1' REFERS TO THE DISK FROM WHICH FILES ARE TO BE COPIED (REMEMBER THAT SOFTLIB'S FULL NAME IS DRA1:[SOFTLIB]). [SOFTLIB.720LDFILE#] IS THE SUB-DIRECTORY FOR THE LOAD FILE RELEASE YOU ARE ARCHIVING.
'MTA0' REFERS TO THE DEVICE TO WHICH FILES ARE TO BE COPIED. 720LDFILE#.BCK IS THE NAME UNDER WHICH THE FILES ARE TO BE STORED ON MAGNETIC TAPE. TO HELP WITH THIS CONCEPT, THINK OF 'BCK' AS STANDING FOR 'BUCKET'. THE MAGNETIC TAPE REEL SCM001 CONTAINS NUMEROUS 'BUCKETS' OR CONTAINERS, EACH OF WHICH CONTAINS ALL THE FILES PERTINENT TO A PARTICULAR LOAD FILE RELEASE. YOU ARE TELLING THE APPROPRIATE SOFTWARE TO COPY FILES FROM A GIVEN DIRECTORY ON VAX TO MTA0, AND TO GIVE THE COLLECTION OF FILES ON MTA0 THE NAME WHICH YOU SPECIFY (NAMELY XXX.BCK).

PRINT THE LIST OF FILES WHICH WERE COPIED TO MAGNETIC TAPE (BY TYPING SHIFT AND PRINT SIMULTANEOUSLY), BOTH FOR YOUR OWN RECORDS AND SO THAT YOU CAN SHOW THE PROJECT ENGINEER THAT THE ARCHIVING WAS INDEED ACCOMPLISHED. (WE DO THIS TO BE NICE, NOT BECAUSE IT'S A REQUIREMENT.)

H. TYPE $DISMOUNT/NOUNLOAD MTA0
THIS REWINDS THE MAGNETIC TAPE WITHOUT ALLOWING THE END OF THE TAPE TO BECOME FREE OF THE TAPE HEAD, AND LOGICALLY DISMOUNTS THE TAPE FROM THE TAPE DRIVE.

I. TYPE $MOUNT MTA0 SCM001
THIS LOGICALLY MOUNTS (THAT IS, CONCEPTUALLY MOUNTS; IT THE THE SOFTWARE WHICH IS BEING MATCHED) THE MAGNETIC TAPE, GIVING IT A STRUCTURE WHICH ALLOWS THE RECOGNITION OF INDIVIDUAL 'BUCKETS'. (WITH THE MOUNT/FOREIGN COMMAND USED BEFORE, NO ATTENTION IS PAID TO ANY INTERNAL STRUCTURE THE TAPE MAY HAVE.)

J. TYPE $DIR MTA0:
THIS WILL RESULT IN A DIPLAY ON THE VAX TERMINAL OF THE 'BUCKETS', OR 720-NUMBERS, CONTAINED ON MTA0 AND THEREFORE ON SCM001. CHECK THAT THE LOAD FILE NUMBER YOU HAVE JUST COPIED FROM VAX IS THERE. HAVE THIS DIRECTORY PRINTED TOO FOR YOUR FUTURE REFERENCE.

K. TYPE $DISMOUNT MTA0
THIS REWINDS THE MAGNETIC TAPE COMPLETELY AND DISCONNECTS OR 'DISMOUNTS' THE APPROPRIATE SOFTWARE THAT YOU HAVE BEEN USING FOR THE TRANSFERRING OF FILES AND FOR THE MANIPULATION OF THE MAGNETIC TAPE.

L. RETURN TO THE COMPUTER ROOM (DON'T FORGET THE NOTE ON YOUR TERMINAL), REMOVE SCM001 FROM THE TAPE DRIVE, REPLACE THE STATIC TAPE AND STYROFOAM ON THE END OF THE MAGNETIC TAPE, PUT SCM001 IN ITS CASE AND LOCK THE CASE, AND RETURN IT TO THE CABINET.

M. RETURN TO YOUR TERMINAL AND TYPE $DEALLOCATE MTA0. THIS ALLOWS OTHER VAX USERS TO USE THE MAGNETIC TAPE DRIVE. YOU ARE NOW IN NORMAL CORRESPONDENCE WITH VAX AND HAVE COMPLETED THE ARCHIVING PROCEDURE FOR YOUR LOAD FILE RELEASE.

16. CHECK FOR 122- PRINT IN BLUEPRINT
COMPONENT ENGINEERING SHOULD HAVE BEEN GENERATING A SPECIFICATION DRAWING FOR THE 122- NUMBER ON THE LOAD FILE RELEASE. GO TO BLUEPRINT AND ASK FOR A COPY OF THE 122- NUMBER. IF THEY ARE UNABLE TO PROVIDE YOU WITH ONE, GO TO COMPONENT ENGINERING AND FIND OUT WHY IT IS NOT YET COMPLETED. DO NOT RELEASE MASTER CHIPS TO COMPONENT ENGINEERING UNTIL THE 122- DRAWING IS AVAILABLE FROM BLUEPRINT.

17. OBTAIN RELEASE SIGNATURES
AT THE BOTTOM OF THE FIRST PAGE OF THE RELEASE ORDER IS A SECTION ENTITLED 'VERIFICATION AND RELEASE AUTHORIZATION' WITH THREE NAMES FILLED IN. YOU ARE NOW READY FOR THE PEOPLE WHOSE NAMES ARE FILLED IN TO INITIAL AND DATE THE BLANKS NEXT TO THEIR NAMES. BE SURE THAT THEY UNDERSTAND THAT THEIR INITIALS CONSTITUTE AN ACCEPTANCE OF RESPONSIBILITY FOR THE SOFTWARE IN THE MASTER CHIPS: THEY ARE STATING, 'I HAVE CHECKED THE MASTER CHIPS, THEY ARE ACCURATE, AND IT IS ALL RIGHT TO USE CHIPS PROGRAMMED FROM THESE MASTERS IN PRODUCTION UNITS'. OBTAIN THE INITIALS OF EACH PERSON LISTED; THEN MAKE THREE COPIES OF THE LOAD FILE RELEASE ORDER AND GIVE A COPY TO EACH OF THE THREE SIGNATORIES.

18. DISTRIBUTE MASTERS
THE DESIGN ENGINEER WILL NORMALLY ALREADY HAVE HIS MASTER CHIP FROM WHEN HE VERIFIED AND VALIDATED IT. TAKE THE OTHER THREE MASTERS TO COMPONENT ENGINEERING AND WATCH THE PERSON WHO ACCEPTS THEM LOG THE FACT OF THEIR RECEIPT IN A THREE-RING BINDER. MAKE SURE THAT THEY WRITE DOWN THE CORRECT PART NUMBER, AND THE CORRECT NUMBER OF MASTERS RECEIVED.

19. COMMENTS

USE THE 'COMMENTS' SECTION OF THE 'LOAD FILE RELEASES IN PROCESS' DOCUMENT (THE RIGHTMOST SECTION AND THE TWO LINES UNDER THE SPACES FOR THE COMPLETION OF EACH STEP) TO RECORD NOTES, PROBLEMS, INSIGHTS, ETC.

20. YOUR FURTHER OBLIGATIONS

ONCE THE LOAD FILE RELEASE IS COMPLETE, YOU ARE STILL RESPONSIBLE FOR GENERATING:

A) 719- AND 722- COVER PAGES TO BE AVAILABLE FROM BLUEPRINT;

B) 718- BIBLIOGRAPHIES IF NONE APPROPRIATE EXISTS FOR THAT LOAD FILE RELEASE;

C) WINSOME REMARKS TO CAUSE THE SOFTWARE PROJECT ENGINEER TO COMPLETE THE 721- DESIGN DESCRIPTION DOCUMENT, HAVE IT TYPED AND APPROVED, AND MAKE IT AVAILABLE FROM BLUEPRINT.

THIS PROCEDURE IS USED AFTER THE LOAD FILES (720) HAVE BEEN RELEASED TO BLUEPRINT FOR DISTRIBUTION AND THERE ARE MANY 720'S IN THE [SOFTLIB] DIRECTORY. THESE 720 DIRECTORIES AND THEIR FILES WILL BE ARCHIVED ON TWO TAPES; SCM01A AND SCM01B. SCM01A IS HELD IN THE SAFE WHILE SCM01B IS HELD IN THE COMPUTER ROOM.

A. BEFORE YOU BEGIN COPYING FILES TO THE MAGNETIC TAPES YOU MUST 'ALLOCATE' THE TAPE DRIVE TO YOURSELF AND THE TERMINAL ON WHICH YOU ARE WORKING. THIS IS TO PREVENT ANYONE ELSE FROM OVERWRITING YOUR PARTICULAR TAPE WHILE YOU ARE USING IT. (IF THEY DID, ALL INFORMATION ON THAT TAPE WOULD BE PERMANENTLY LOST.) IN ORDER TO 'ALLOCATE' THE TAPE DRIVE (AFTER LOGGING ON) TYPE

```
$ ALLOCATE MTA0:
```

MTA0 STANDS FOR 'MAGNETIC TAPE NUMBER A0' AND MEANS THAT YOU ARE DESIGNATING A SPECIFIC TAPE DRIVE FOR YOUR USE ONLY.

B. LEAVE A NOTE ON THE TERMINAL THAT SAYS YOU WILL BE RIGHT BACK FROM CHECKING ON THE TAPE DRIVE. THIS IS SO SOMEONE DOESN'T LOG YOU OFF, THEREBYE DEALLOCATING THE TAPE DRIVE AND CREATING THE POSSIBILITY OF SOMEONE ELSE USING MTA0 AND ERASING YOUR TAPE.

C. GO TO THE COMPUTER ROOM AND GET THE SOFTWARE CONTROL BACKUP TAPE MARKED SCM01B FROM THE TAPE CABINET. (IF THE CABINET IS LOCKED HAVE ONE OF THE COMPUTER PERSONNEL UNLOCK IT FOR YOU.) THE TAPE IS ON A SMALL REEL.

D. LOCATE THE TAPE DRIVE CONNECTED TO SILVER. MAKE SURE THAT THE 'BOT' AND 'ON LINE' LIGHTS ARE OFF. OPEN THE PLASTIC RING THAT ENCIRCLES THE REEL AND REMOVE THE STATIC TAPE FROM THE TAPE. LOCATE AND INSERT THE 'WRITE RING' INTO THE BACK SIDE OF THE TAPE REEL. (IF THIS IS NOT DONE, THE COMPUTER WILL NOT BE ABLE TO WRITE ANYTHING ONTO THE TAPE.) OPEN THE TAPE DRIVE DOOR AND LOAD THE REEL WITH THE MAGNETIC TAPE COMING OVER THE REEL FROM THE LEFT. LOCK THE REEL IN PLACE BY PRESSING IN THE RAISED SIDE OF THE BLACK PLASTIC LOCK IN THE CENTER OF THE THE TAPE HUB. INSERT ABOUT THREE TO FIVE INCHES OF TAPE INTO THE FEED MECHANISM.

E. CLOSE THE TAPE DRIVE DOOR. PRESS THE LOAD/REW BUTTON TO LOAD THE TAPE. WHEN THE TAPE HAS FINISHED LOADING AND THE 'BOT' LINE HAS COME ON PRESS THE 'ON LINE' BUTTON. CHECK THAT BOTH THE 'BOT' AND 'ON LINE' LIGHTS ARE ON. THE TAPE DRIVE IS ON AND READY.

F. RETURN TO YOUR TERMINAL. TYPE

```
$MOUNT/FOREIGN MTA0:
```

THIS LOADS OR MOUNTS THE APPROPRIATE SOFTWARE (LOGICALLY AND PHYSICALLY) WITH THE APPROPRIATE FORMAT TO ALLOW YOU TO TRANSFER FILES FROM VAX TO THE END OF THE MAGNETIC TAPE (AREA JUST AFTER THE LAST ENTRY).

G. CONNECT YOUR TERMINAL TO THE PRINTER AND PRINT A COPY OF THE [SOFTLIB] DIRECTORY. GO THROUGH EACH 720 DIRECTORY AND PURGE THE FILES OF ALL UNNECESSARY MATERIALS.

H. EDIT THE SCM01B.COM FILE IN THE [SOFTLIB] DIRECTORY TO REFLECT ALL THE 720 DIRECTORIES THAT YOU WANT TO PUT ON THE SCM01B TAPE. AS YOU INSERT THE NEW NUMBERS INTO THE SCM01B.COM FILE REMOVE THOSE LISTED THAT HAVE ALREADY BEEN PUT ON THE TAPE IN THE PREVIOUS SESSION.

I. WHEN YOUR EDITING SESSION IS COMPLETE AND THE SCM01B.COM CONTAINS ALL THE 720 NUMBERS THAT YOU WANT ARCHIVED TYPE

@SCM01B.COM

AT THE SAME TIME PRESS CTRL/PRINT TO RECEIVE A PRINTOUT OF THE NEWLY ARCHIVED DIRECTORIES AND FILES.

J. WHEN THE $ PROMPT RETURNS TYPE

$DISMOUNT/NOUNLOAD MTA0:

THIS WILL REWIND THE TAPE WITHOUT TAKING IT OFF-LINE OR UNLOADING IT. IT IS SIMILAR TO A REWIND COMMAND BUT THE USER MUST ISSUE A MOUNT COMMAND BEFORE USING THE TAPE AGAIN.

K. TYPE

$ MOUNT MTA0: SCM01B

THIS MOUNTS THE TAPE VOLUME ON DRIVE MTA0.

L. TYPE

$ DIR MTA0:

THIS WILL LIST ON HARD COPY ALL THE 720 DIRECTORIES THAT ARE NOW CONTAINED ON THE SCM01B TAPE.

M. TYPE

$TIME

THIS WILL GIVE THE EXACT TIME AND DATE ON THE HARDCOPY TO DETERMINE EXACTLY WHEN THE DIRECTORIES ON THIS TAPE WERE ARCHIVED.

N. TYPE

$SHOW PROC

THIS WILL SHOW THE DIRECTORIES USED, THE DEVICES ALLOCATED, AND WILL PROVIDE OTHER INFORMATION ON HARDCOPY TO BE USED IN CASE THE FILE EVER HAS TO BE THOROUGHLY LOOKED AT FOR SOME TYPE OF ERROR OR PROBLEM.

O. TYPE

$ DISMOUNT MTA0:

THIS COMMAND WILL REWIND THE TAPE, TAKE IT OFF-LINE, AND WILL UNLOAD IT FROM THE TAPE DRIVE.

P. GO TO THE SAFE AND LOCATE THE TAPE MARKED SCM01A. LEAVE A NOTE ON YOUR TERMINAL INDICATING WHERE YOU ARE AND GO BACK TO THE COMPUTER ROOM. REMOVE SCM01B FROM THE TAPE DRIVE, REMOVE THE WRITE RING, INSTALL THE WRITE RING ON SCM01A, MOUNT SCM01A THE SAME WAY THAT YOU MOUNTED SCM01B, AND PUT SCM01B AND THE NEW HARDCOPY OF THE .COM FILE BACK INTO THE CABINET. GO BACK TO YOUR TERMINAL AND REPEAT STEPS "D." THROUGH "O." SUBSTITUTING SCM01A WHERE SCM01B APPEARS.

Q. GO TO THE COMPUTER ROOM AFTER LEAVING A NOTE ON YOUR TERMINAL EXPLAINING WHERE YOU ARE. REMOVE SCM01A FROM THE TAPE DRIVE, REMOVE THE WRITE RING, REPLACE THE STATIC TAPE ON THE END OF THE MAGNETIC TAPE, REINSTALL THE OUTER PLASTIC RING ON THE TAPE REEL, AND RETURN THE TAPE AND THE NEW HARDCOPY PRINTOUT TO THE SAFE.

R. RETURN TO YOUR TERMINAL AND TYPE

$ DEALLOCATE MTA0:

THIS WILL DEALLOCATE THE TAPE DRIVE MTA0 SO THAT IT MAY BE USED BY OTHERS.

S. TAKE THE OLD HARDCOPIES OF THE SCM01A AND SCM01B PRINTOUTS AND FILE THEM IN THE FOLDER MARKED SCM01A AND SCM01B HARDCOPY PRINTS.

THIS CONCLUDES YOUR ARCHIVING PROCEDURES FOR THE 720 LOAD FILES.

MEMO TO: KENT QUICK DATE: 1 NOV 1982

FROM: STEVE RUSSELL

SUBJECT: PREPARATION AND MAINTANENCE OF SOFTWARE CONTROL DOCUMENTS

IT NOW LOOKS LIKE WE WILL NEED SOME SPECIAL HANDLING OF THE SOFTWARE DOCUMENTS PRODUCED FOR THE SOFTWARE CONFIGURATION MANAGEMENT SCHEME THAT WE ARE CURRENTLY IMPLEMENTING. TO EXPLAIN THE PROPOSED METHOD, I WILL USE THE DESIGN DESCRIPTION DOCUMENT AS AN EXAMPLE. THIS DOCUMENT WILL HAVE TEXT, LARGE TABLES, AND FLOW CHARTS. THE TEXT, AS SUBMITTED BY THE ENGINEER, WILL BE ON SIZE A PAPER BUT THE TABLES AND FLOW CHARTS WILL LIKELY BE ON BOTH SIZE A AND SIZE B. ALSO THE QUALITY OF THE TABLES AND CHARTS WILL NOT BE SUITABLE FOR DIRECT SUBMITTAL TO THE PRINT ROOM BUT WILL NEED TO BE CLEANED UP.

I AM MAKING THE FOLLOWING RECOMMENDATIONS FOR PREPARATION OF THIS AND SIMILIAR SOFTWARE DOCUMENTS:

1. THE DESIGN ENGINEER CAN SUBMIT TEXT, FIGURES(FLOW CHARTS), AND TABLES, TO JAN O'DELL FOR PROCESSING. THE PROCEDURE CAN BE THE SAME AS THAT USED FOR WORD PROCESSING.
2. KENT QUICK CAN EVALUATE THE QUALITY OF THE FIGURES AND TABLES AND HAVE DRAFTING PREPARE THOSE THAT ARE NEEDED.
3. REQUIRE THAT ALL FIGURES BE EITHER A OR B SIZE.
4. HAVE WORD PROCESSING PREPARE THE TEXT.
5. ASSIGN A B SIZE FILING CABINET IN THE PRINT ROOM THAT IS RESERVED FOR SOFTWARE DOCUMENTS AND FILE THE A AND B SIZE PAGES OF A SINGLE REPORT TOGETHER.
6. IMPLEMENT A SCHEME OF RESTRICTING ACCESS TO COMPANY CONFIDENTIAL DOCUMENTS SO THAT A PROPERLY PREPARED AND SIGNED REQUEST IS NEEDED TO PRODUCE A COPY.
7. ECO OF THE DOCUMENT CAN BE DONE IN THE SAME WAY AS OTHER WORD PROCESSED DOCUMENTS ARE.

STEVE RUSSELL

COPIES: GARY BURRELL
KEVIN SMITH
JIM MOSS
FRANK KURTZ

NAME: STEVE RUSSELL
DATE: 22 DEC 1982
REV: 18 APRIL 1983

[SOFTLIB.DOC]BACKFILL.LIS

PROCEDURE FOR RELEASING SOFTWARE INTO CONFIGURATION MANAGEMENT AT THE END OF A PROJECT CYCLE.

[SOFTLIB.DOC]BACKFILL.LIS

FROM: STEVE RUSSELL
DATE: 22 DEC 1982
REV: 18 APRIL 1983

---

1. OBTAIN A LOAD FILE SOFTWARE RELEASE FORM FROM SOFTWARE CONTROL.

2. OBTAIN 5 COPIES OF THE SOFTWARE DOCUMENT NUMBER REQUEST FORM FROM JAN O'DELL AND APPLY FOR ASSIGNMENTS OF THE FOLLOWING NUMBERS:
   - 715-XXXX-RN -- LRU SOFTWARE DOCUMENT.
   - 719-XXXX-RN -- SOURCE LISTING DOCUMENT.
   - 720-XXXX-RN -- LOAD FILE RELEASE DOCUMENT.
   - 721-XXXX-RN -- DESIGN DESCRIPTION DOCUMENT.
   - 722-XXXX-RN -- MACHINE CODE DOCUMENT. (THIS REQUEST MUST ALSO BE SIGNED BY SOFTWARE CONTROL BEFORE THE NUMBER CAN BE ASSIGNED)

   BE SURE TO USE DESCRIPTIVE NAMES IN THE STANDARD DESCRIPTION FIELD AND NOT GENERIC NAMES SUCH AS MICROPROCESSOR OR LOAD FILE. TRY TO HAVE DESCRIPTIONS THAT WILL RELATE TO YOUR PROJECT. NOTE THAT NONE OF THE PART NUMBERS WILL BE KEYED (SAME CENTER DIGITS) AS IS CURRENTLY DONE FOR HARDWARE.

3. AFTER THE 722 NUMBER HAS BEEN ASSIGNED, APPLY TO COMPONENT ENGINEERING (P.J. OR DALE COOPER) FOR A PROGRAMMED DEVICE PART NUMBER 122-XXXX-RN. THIS IS THE NUMBER THAT WILL BE PUT ON THE BILL-OF-MATERIALS. YOU WILL BE REQUIRED TO FILL OUT A FORM GIVING THE 722 NUMBER, THE GENERIC PART NUMBER (120-XXXX-XX) OF THE ELECTRICAL PART TO BE PROGRAMMED, AND THE EXPECTED RELEASE DATE OF THE FINISHED SOFTWARE.
   NOTE!: THE 122 PRINT SHOULD BE RELEASED AND IN THE PRINT ROOM PRIOR TO THE PURCHASING RELEASE SO THAT IT CAN BE PULLED BY PURCHASING AND THE CORRECT UNPROGRAMMED CHIPS CAN BE ORDERED.

4. AFTER THE 122 NUMBER HAS BEEN ASSIGNED, YOU WILL BE ABLE TO FILL OUT THE LOAD FILE RELEASE FORM OBTAINED IN STEP #1.
   A) ENTER THE LOAD FILE SOFTWARE DOCUMENT NUMBER (720) AT THE BOTTOM OF EACH PAGE.
   B) ENTER THE SYSTEM NAMES OF ALL SYSTEMS THAT WILL BE USING THIS PROGRAMMED DEVICE. GENERALLY ONLY ONE SYSTEM NAME WILL BE ENTERED.
   C) ENTER THE NAMES OF ALL UNITS (LRUS) THAT WILL BE USING THE PROGRAMMED DEVICE. FOR EACH DIFFERENT UNIT, A DIFFERENT 715 NUMBER IS REQUIRED UNLESS ALL OF THE SOFTWARE IN EACH DIFFERENT UNIT IS IDENTICAL.
   D) ENTER THE PROCESSOR NAME. THIS SHOULD BE A UNIQUE AND DESCRIPTIVE NAME THAT APPLIES TO THE APPLICATION OR FUNCTION TO BE PERFORMED BY THE PROCESSOR. IT IS USED FOR DESCRIPTIVE PURPOSES IN THE APPLICABLE SOFTWARE DOCUMENTS.
   E) ENTER THE VAX FILE NAMES FOR THE COMMAND FILE, LOAD FILE, SOURCE FILE, AND OBJECT FILE. TRY TO USE A UNIQUE AND DESCRIPTIVE NAME AND ALSO PROVIDE FOR A METHOD OF NOTING THE REVISION LEVEL OF THE SOURCE. FOR EXAMPLE, THE NAMES FOR THE KAC-952 ANTENNA COUPLER PROCESSOR COULD BE KAC952V01.COM, KAC952V01.LOA, KAC952V01.I48, & KAC952V01.LOA. IN THIS CASE, THE SOURCE FILE IS .I48 AND THE VERSION LEVEL (V01) IS PART OF THE VAX FILENAME.

F) ENTER THE NAMES OF THE SOFTWARE PROJECT ENGINEER AND THE ENGINEERING GROUP LEADER. THESE ARE THE PEOPLE THAT WILL AUTHORIZE THE ACTUAL RELEASE OF MASTER CHIPS TO MANUFACTURING.

G) ENTER THE ASSIGNED PART NUMBERS ON THE SECOND PAGE WHERE INDICATED. IF THE RELEASE IS FOR A 16-BIT PROCESSOR, THE EVEN AND ODD BYTE VAX FILE NAMES MUST ALSO BE ASSIGNED. THE LAST SIX DIGITS OF THE 722 DOCUMENT NUMBER ARE ALSO USED IN THE DOCUMENT FILE NAME AND ASCII-HEX FILE NAME.

H) INDICATE THE PROGRAMMED DEVICE TYPE IN THE BLANK PROVIDED. THIS IS USED BY SOFTWARE CONTROL AS A CROSSCHECK TO MAKE SURE THE CORRECT TARGET DEVICE IS USED TO MAKE MASTER COPIES.

I) INDICATE THE SUPPORT SOFTWARE VERSIONS THAT YOU WANT USED IN THE RELEASE. THESE SHOULD AGREE WITH THE VERSIONS THAT ARE SPECIFIED IN THE .COM FILE REFERENCED ON THE FIRST PAGE. EVENTUALLY THIS INFORMATION WILL BE INCLUDED IN A SYSTEM HELP COMMAND TO ASSIST THE USERS.

K) ENTER ASSIGNED NUMBERS FOR SOURCE LISTING AND DESIGN DESCRIPTION. OBTAIN PROGRAMMER'S LITERATURE NUMBER FROM SOFTWARE CONTROL. REFERENCE BILL-OF-MATERIAL ASSEMBLY NUMBER AND INCLUDE THE ECO NUMBER IF THE ACTION IS A CHANGE TO SOFTWARE ALREADY RELEASED.

5. AFTER THE FORM HAS BEEN FILLED OUT, RETURN IT TO SOFTWARE CONTROL. THE INFORMATION PROVIDED ON THE LOAD RELEASE FORM WILL NOW BE USED TO CREATE THE NECESSARY LOAD FILES AND DOCUMENT FILES ON VAX. THE PROCEDURE USED INSURES THAT ALL SOFTWARE AND SUPPORT DOCUMENTS WILL BE ARCHIVED AND THAT THE LOAD FILES CAN BE RECREATED IN THE FUTURE WITH THE IDENTICAL SOURCE FILE AND SOFTWARE SUPPORT ENVIRONMENT.

6. THE SOURCE AND LOAD FILES WILL BE PLACED IN THE [EXCHANGE] DIRECTORY AND YOU WILL BE ASKED TO DO A DIRECT COMPARISION OF THE CODE GENERATED IN THE RELEASE WITH THE CODE YOU ARE USING IN THE CURRENT ENGINEERING MODEL. THIS STEP ELIMINATES OBVIOUS OR GROSS ERRORS PRIOR TO THE CREATION OF MASTER CHIPS.

7. AFTER YOU HAVE VERIFIED THAT THE CODE IS CORRECT ON VAX, SOFTWARE CONTROL WILL AGAIN USE THE INFORMATION PROVIDED ON THE LOAD RELEASE FORM TO CREATE FOUR MASTER COPIES OF THE PROGRAMMED DEVICE. THESE MASTERS ARE USED BY MANUFACTURING TO GANG PROGRAM PRODUCTION CHIPS. THEY ARE ALSO USED IN QUALITY CONTROL PROCEDURES SUCH AS THE VERIFICATION AND VALIDATION OF THE PROGRAMMED DEVICE BY THE SOFTWARE PROJECT ENGINEER. SOMETIMES YOU MAY BE ASKED TO PROVIDE ERASED DEVICES FOR THIS PURPOSE.

8. AFTER A MASTER CHIP SET HAS BEEN PRODUCED, A SINGLE COPY WILL BE RETURNED TO THE DESIGN GROUP FOR VERIFICATION AND VALIDATION. THIS INVOLVES A BIT-BY-BIT VERIFICATION OF THE MACHINE CODE IN THE MASTER CHIP AGAINST THE CORRESPONDING CHIP IN THE ENGINEERING PROTOTYPE. THIS IS USUALLY DONE ON A PROM PROGRAMMER WITH A VERIFY OR COMPARE COMMAND. IN ADDITION, THE MASTER CHIP SHOULD BE USED IN THE PROTOTYPE TO VALIDATE ALL SOFTWARE FUNCTIONS IMPLEMENTED WITH THIS PARTICULAR CHIP.

9. AFTER VERIFICATION AND VALIDATION, THE SOFTWARE PROJECT ENGINEER AND THE ENGINEERING GROUP LEADER WILL SIGN AND DATE THE LOAD FILE RELEASE FORM. THIS SIGNIFIES THAT THE SOFTWARE IS CORRECT SO IT CAN BE ARCHIVED ON MAGNETIC TAPE AND RELEASED TO MANUFACTURING. THE DESIGN GROUP KEEPS THEIR MASTER COPY AS A READY REFERENCE TO THE ACTUAL CODE IN THE RELEASED CHIP. AT THIS TIME, SOFTWARE CONTROL WILL ARCHIVE ALL VAX FILES ASSOCIATED WITH THIS RELEASE AND WILL SUBMIT THE REMAINING THREE MASTER COPIES TO COMPONENT ENGINEERING.

NOTE: WE CURRENTLY PLAN TO COMPUTERIZE THESE FORMS SO THE DESIGNER CAN SIT AT A TERMINAL AND RECEIVE ALL THE NECESSARY PROMPTS AND INSTRUCTIONS. UNTIL THIS CAN BE IMPLEMENTED, WE WILL CONTINUE WITH THE CURRENT MANUAL PROCEDURE.

PROCEDURE FOR PMDN OF 122 CHIPS

A MATERIAL DEVIATION FOR PROGRAMMED MEMORY CHIPS (FIRMWARE) CAN BE USED IN EMERGENCY SITUATIONS TO SUBSTITUTE A BASE ELECTRICAL PART THAT IS BETTER THAN THE ONE LISTED ON THE 122 PRINT OR TO INSTALL MODIFIED SOFTWARE PRIOR TO ACCEPTANCE TESTING AND RELEASE OF THE NEW VERSION. THE PMDN (PRODUCTION MATERIAL DEVIATION NOTICE) SHOULD BE USED VERY SPARINGLY FOR SOFTWARE BECAUSE OF THE RISKS ASSOCIATED WITH LACK OF ACCEPTANCE TESTING. A SOFTWARE PMDN SHOULD BE RESTRICTED TO QUANTITIES OF LESS THAN TWENTY PIECES.

FIRMWARE PARTS INSTALLED UNDER A PMDN SHALL BE LABELED AS FOLLOWS;

```
**************************
*                        *
*       PMDN-YYYYY       *
*                        *
*       122-XXXX-RN      *
*                        *
**************************
```

WHERE:

1. YYYYY IS THE PMDN NUMBER.
2. 122-XXXX-RN IS THE PART NUMBER OF THE FIRMWARE PART BEING REPLACED BY THE DEVIATED PART. THE REVISION NUMBER IS NOT THE NEW (ANTICIPATED) NUMBER BUT THE ONE CURRENTLY SPECIFIED ON THE BILL OF MATERIALS.

COMMON PRACTICE WILL DICTATE THAT PARTS SO MARKED WITH A PMDN NUMBER FOR SOFTWARE CHANGE REASONS WILL NOT BE TRACEABLE AND NOT BE SUPPORTABLE.

SOFTWARE METHODOLOGY
SILVER::ENG:[SOFTLIB.DOC]
CH25.LIS

STEVE RUSSELL
19 APRIL 1983
REV: 20 JUNE 1983

**NOTE: THIS CHAPTER WILL INITIALLY BE IN THE FORM OF A MEMO AND LATER WILL BE EDITED TO INCLUDE JUST THE INFORMATION.

DATE: 19 APRIL 1983
20 JUNE 1983

TO: DISTRIBUTION

FROM: KOREY WILLNAUER X2704
STEVE RUSSELL X2505

SUBJECT: PROCEDURE FOR APPROVING EPROM CHANGE TO MASKED ROM.

IN THE PAST SEVERAL MONTHS THERE HAVE BEEN MANY DISCUSSIONS ABOUT HOW TO DETERMINE WHEN TO CHANGE FROM AN ELECTRICALLY-PROGRAMMED PART TO A MASKED-PROGRAMMED PART. PURCHASING, COMPONENT ENGINEERING, AND DESIGN ENGINEERING ALL FEEL THAT THERE HAS NEVER BEEN A CLEARLY ESTABLISHED PROCEDURE FOR MAKING THIS DECISION. THE ENCLOSURE WITH THIS MEMO OUTLINES WHAT WE FEEL IS A REASONABLE PROCEDURE AND EXPLAINS THE FORM THAT HAS BEEN DEVELOPED.

DISTRIBUTION:

PROCEDURE FOR APPROVING EPROM CHANGE TO MASKED ROM

INTRODUCTION

THIS CHAPTER OUTLINES THE DECISION MAKING PROCESS FOR CHANGING AN EPROM MEMORY PART TO A MASKED ROM PART. THE PROCEDURE THAT HAS BEEN ESTABLISHED HAS COME OUT OF SEVERAL DISCUSSIONS WITH PURCHASING, COMPONENT ENGINEERING, DESIGN ENGINEERING, AND SOFTWARE CONTROL. AN OVERVIEW OF THE DECISION MAKING PROCESS WILL BE GIVEN FIRST AND FOLLOWED BY A DETAILED DESCRIPTION OF HOW TO PROCESS THE FORM THAT WILL BE USED.

THE KEY PARAMETERS AFFECTING THIS DECISION THAT HAVE BEEN IDENTIFIED ARE:

1. QUANTITY USED.
2. PRICE OF EPROM PART VS MASKED PART.
3. STABILITY OF THE CODE.
4. SPECIFICATION OF A SUITABLE BASE ELECTRICAL PART.

THE PROCESS TAKES THESE INTO CONSIDERATION.

THE FLOWCHART OF FIGURE 25.1 ILLUSTRATES THE BASIC STEPS IN THE DECISION MAKING PROCESS. FIRST, PURCHASING EVALUATES THE NEED TO CHANGE TO A MASKED PART BY DETERMINING CURRENT AND PROJECTED USAGE. THE MONTHLY REQUIREMENTS PLANNING REPORT (RPR) CAN BE USED TO MAKE THESE ESTIMATES. THE DETERMINATION OF WHAT CONSTITUTES A SUFFICIENTLY HIGH LEVEL OF USAGE MUST BE MADE SUBJECTIVELY. AS EXPERIENCE IS GAINED WITH THIS PROCESS, IT WILL BE EASIER TO TELL WHEN TO INITIATE IT.

IF QUANTITIES JUSTIFY, PURCHASING INITIATES THE FORM AND ROUTES IT TO THE APPROPRIATE DESIGN ENGINEERING GROUP. THE GROUP EVALUATES THE MATURITY OF THE SOFTWARE DESIGN. IF THEY FEEL THAT IT IS MATURE ENOUGH FOR MASKING, THEY SEND THE FORM TO COMPONENT ENGINEERING FOR EQUIVALENCE TYPING. IF IT IS NOT YET MATURE ENOUGH, THE FORM IS RETURNED TO PURCHASING.

COMPONENT ENGINEERING EVALUATES THE PRESENTLY USED EPROM PART AND WORKS WITH THE DESIGN GROUP TO RECOMMEND AN EQUIVALENT MASKED PART. WHEN THE BEST REPLACEMENT PART HAS BEEN IDENTIFIED, THE FORM IS RETURNED TO PURCHASING SO THEY MAY GATHER COST DATA AND DO A TWELVE-MONTH COST ANALYSIS. USING THE COST ANALYSIS, PURCHASING EVALUATES THE DATA AND CONSULTS WITH THE DESIGN GROUP TO MAKE A FINAL DECISION.

FOR A YES DECISION, PURCHASING INITIATES THE PROCEDURE FOR GETTING A MASKED PART MADE AND STOCKED. DETAILS OF THIS PROCEDURE ARE CONTAINED IN A SUBSEQUENT CHAPTER.

DESCRIPTION OF FORM

TOP OF FORM. (PURCHASING)

THE INITIATOR IDENTIFIES THE PROGRAMMED EPROM PART NUMBER AND THE ENGINEERING GROUP THAT HAS DESIGN RESPONSIBILITY FOR THE PRODUCT. IF THE ENGINEERING GROUP IS NOT KNOWN, THE UNIT NAME LISTED ON THE RPR CAN BE USED BY DAN HARDER OR FRANK KURTZ TO IDENTIFY THE GROUP.

SECTION I. INITIATION (PURCHASING)

USING THE INFORMATION AVAILABLE IN THE CURRENT RPR, DETERMINE THE MPO SCHEDULE AND PRESENT PIECE PART COST OF THE EPROM AND RECORD THEM. REMEMBER THAT THE DESIRED NUMBER IS THE AVERAGE PER MONTH AS SCHEDULED OVER A TWELVE (12) MONTH PERIOD. NEXT, SIGN AND DATE THE FORM AND SEND IT TO THE COGNIZANT ENGINEERING GROUP.

SECTION II. CODE STABILITY EVALUATION (DESIGN ENGINEERING)

IT IS VERY IMPORTANT THAT THE CODE BE EVALUATED FOR STABILITY BEFORE A DECISION TO CONVERT TO A MASKED PART IS MADE. STABILITY

EVALUATION IS A COMPLEX PROCESS INVOLVING LIMITED FIELD DATA AND A LOT OF INTUITIVE JUDGEMENT. THIS ACTIVITY CAN BEST BE DONE BY THE DESIGN GROUP RESPONSIBLE FOR THE UNIT.

TO ASSIST IN THIS DECISION, INFORMATION CAN BE GATHERED FROM PRODUCT SUPPORT AND THE ECO REVISION HISTORY. IF THERE HAVE BEEN COMPLAINTS FROM PRODUCT SUPPORT ABOUT FIELD PROBLEMS WITH THE UNIT SOFTWARE, IT MAY BE DESIRABLE TO POSTPONE MASKING OR TO CONSIDER MAKING A REVISION BEFORE MASKING. LIKEWISE, IF THE ECO REVISION HISTORY INDICATES FREQUENT CHANGES ARE STILL BEING MADE, IT IS PROBABLE THAT THE CODE IS NOT YET STABLE ENOUGH FOR MASKING.

HOWEVER, IF THE FIELD HISTORY IS GOOD AND IT HAS BEEN A "LONG TIME" SINCE THE LAST REVISION (USE YOUR OWN JUDGEMENT), IT IS LIKELY THAT THE STABILITY IS GOOD ENOUGH TO SUPPORT A DECISION TO MASK THE PART.

INDICATE A YES OR NO DECISION ON STABILITY, RECORD THE DATE OF THE MOST RECENT ECO REVISION TO THE LOAD FILE, AND OBTAIN THE GROUP LEADER'S SIGNATURE AUTHORIZING THE STABILITY DECISION. IF THE ANSWER IS YES, SEND THE FORM TO COMPONENT ENGINEERING. IF IT IS NO, INDICATE THE DATE YOU FEEL THE SOFTWARE MIGHT BECOME STABLE ENOUGH TO DO A RE-EVALUATION AND RETURN THE FORM TO THE INITIATOR IN PURCHASING.

SECTION III. BASE PART SELECTION (COMPONENT ENGINEERING)

USING INFORMATION FROM THE 120 SPECIFICATION REFERENCED ON THE 122 PRINT, COMPONENT ENGINEERING CAN SELECT A SUITABLE BASE ELECTRICAL PART THAT CAN BE USED AS A MASK-PROGRAMMED SUBSTITUTE FOR THE EPROM PART. MOST OF THE TIME, PURCHASING CAN OBTAIN A GOOD QUOTE USING ONLY A GENERIC PART NUMBER (I.E. 8048, 8051, Z8000, ETC.) PLUS THE REQUIRED SPEED AND TEMPERATURE RANGE. IN SOME CASES, HOWEVER, DESIGN ENGINEERING MIGHT HAVE SOME UNIQUE REQUIREMENTS THAT MUST BE MET. THE COMPONENT ENGINEER SHOULD CONTACT THE DESIGNER TO VERIFY THAT THE BASE PART WILL MEET HIS DESIGN NEEDS.

THE INFORMATION ENTERED FOR THE BASE PART WILL VARY. USUALLY, THE GENERIC NUMBER IS USED BUT SOMETIMES A BASE PART IS ALREADY IN THE KING RADIO DESCRIPTION MASTER AND THE PRINT IS AVAILABLE TO SEND TO THE VENDOR. THE APPLICABLE 120 NUMBER SHOULD BE ENTERED WHEN THIS IS THE CASE. THE BASE PART COULD ALSO BE DESCRIBED BY THE ORDER NUMBER FOR A SINGLE VENDOR.

SPACE HAS BEEN PROVIDED FOR COMPONENT ENGINEERING TO RECOMMEND POSSIBLE VENDORS.

SIGN AND DATE THE FORM AND SEND IT TO THE INITIATOR IN PURCHASING.

SECTION IV. TWELVE-MONTH COST ANALYSIS (PURCHASING)

WITH THE QUANTITY AND BASE PART INFORMATION NOW AVAILABLE, PURCHASING CAN OBTAIN THE DESIRED COST DATA FROM THE VENDORS AND CALCULATE THE TWELVE-MONTH COST FOR MASKED ROMS AND COMPARE IT WITH THE TWELVE-MONTH COST OF THE SAME NUMBER OF EPROMS. AN OUTLINE OF THE PROCEDURE IS AS FOLLOWS.

CONTACT QUALIFIED VENDORS TO OBTAIN QUOTES ON MASKED PARTS. THIS CAN BE DONE VERBALLY TO HELP SPEED THE PROCESS BUT WRITTEN CONFIRMATION SHOULD ALSO BE REQUIRED. IN ADDITION TO THE BASE PART NUMBER, BE SURE TO INCLUDE ANY SPECIAL REQUIREMENTS (SPEED, TEMPERATURE RANGE, MILITARY SPECS, ETC.) THAT HAVE BEEN REQUESTED BY THE DESIGN GROUP OR COMPONENT ENGINEERING. REQUEST THAT THE VENDOR SUPPLY INFORMATION ON THE MASK CHARGE, MINIMUM ORDER QUANTITY, AND COST PER PART.

USING THE DATA FOR EACH VENDOR, CALCULATE THE TWELVE-MONTH COST FOR EACH AND INSERT THE RESULTS FOR THE LOWEST COST VENDOR IN THE SPACES PROVIDED UNDER THE HEADING "INITIAL COST OF ROM". THE AVG. MPQ NUMBER IS THE SAME AS THE NUMBER REPORTED IN THE MPQ SCHEDULE IN SECTION I. IF YOU FIND THAT MULTIPLYING 12 TIMES THE AVG. MPQ IS LESS THAN THE MINIMUM ORDER QUANTITY, THEN MULTIPLY THE MINIMUM ORDER QUANTITY BY THE PRICE EA. AND ADD THE MASK CHARGE TO FIND THE TWELVE-MONTH COST FOR USING MASKED ROM PARTS IN THE UNIT.

THE COST OF CONTINUING TO USE AN EPROM PART IS CALCULATED IN A SIMILIAR WAY EXCEPT THERE IS NO MASKING CHARGE. THE PIECE PART COST CAN BE DETERMINED FROM THE LAST PURCHASE OF SIMILIAR QUANTITIES.

THE COST SAVINGS FOR THE TWELVE MONTH PERIOD IS COMPUTED BY SUBTRACTING THE TWELVE-MONTH COST OF THE MASKED ROM PART FROM THE TWELVE-MONTH COST OF THE EPROM PART.

SECTION V. DECISION (PURCHASING)

THE FINAL DECISION TO CREATE A MASKED ROM PART IS MADE BY PURCHASING WITH INPUTS FROM DESIGN ENGINEERING. THE DECISION IS PROBABLY NEVER EASY UNLESS THE COST SAVINGS ARE CONSIDERABLE. FOR EXAMPLE, THE COST ADVANTAGE OF USING MASKED PARTS MAY BE SMALL ENOUGH THAT THE DESIGN GROUP DOES NOT FEEL THE SAVINGS JUSTIFY THE RISK OF A SOFTWARE CHANGE. IN OTHER CASES, THE MASKED PARTS MAY BE MORE COSTLY OVER THE NEXT YEAR BUT THE TREND IN SALES AND THE MPO INDICATE INCREASING PRODUCTION QUANTITIES OVER AN 18 OR 24 MONTH PERIOD WILL RESULT IN A SAVINGS. HOPEFULLY, EXPERIENCE GAINED USING THIS PROCEDURE WILL MAKE THE DECISIONS EASIER.

IF A DECISION IS MADE TO GO AHEAD AND CREATE A MASKED PART, CHECK THE YES BOX AND SIGN AND DATE THE FORM. SEND COPIES TO COMPONENT ENGINEERING AND THE DESIGN ENGINEER AND INITIATE THE PROCEDURE FOR OBTAINING THE MASKED ROM.

IF THE DECISION IS NO, RECOMMEND A DATE FOR FUTURE RE-EVALUATION AND FILE THE REQUEST FORM IN PURCHASING FOR FUTURE REFERENCE.

****

* REQUEST FOR MASK-PROGRAMMED COMPONENT *

****

PRESENT EPROM PART NUMBER: 122-_____-____.

COGNIZANT ENGINEERING GROUP: _____.

I. INITIATION (PURCHASING)

MPQ SCHEDULE (AVG. PER MO.) PRESENT COST EA.

INITIATED BY: _____ (SIGNATURE) DATE: _____

(FORWARD THIS FORM TO COGNIZANT ENGINEERING GROUP)

II. CODE STABILITY EVALUATION (DESIGN ENGINEERING)

DATE OF LAST REVISION TO SOFTWARE: _____.

THE CODE FOR THIS PRODUCT WAS EVALUATED TO SEE IF IT IS MATURE AND STABLE ENOUGH FOR MASKING. DATA FROM FIELD EVALUATION AND THE ECO HISTORY INDICATE THE FOLLOWING DECISION:

* YES, THE PART SHOULD BE MASKED.

** NO, RE-EVALUATE ON _____ (DATE).

GROUP LEADER: _____ (SIGNATURE) DATE: _____.

*(IF YES, FORWARD REQUEST TO COMPONENT ENGINEERING.)
**(IF NO, FORWARD THIS FORM TO PERSON INITIATING THIS REQUEST.)

III. BASE PART SELECTION (COMPONENTS ENGINEERING)

BASE PART NO. SPEED TEMP. RANGE

SUGGESTED VENDORS: _____

_____

_____

COMPONENT ENGINEER: _____ (SIGNATURE) DATE: _____

(FORWARD THIS FORM TO PURCHASING INITIATOR.)

IV. TWELVE-MONTH COST ANALYSIS (PURCHASING)
(QUOTE FROM MANUFACTURER)

BASE PART NO.	MASK CHARGE	MIN. QUANTITY	PRICE EA.

(INITIAL COST OF ROM)

12	X	AVG. MPO	X	PRICE EA.	+	MASK CHARGE	=	12 MO. COST FOR MASKED ROM
1 YR.								

(NOTE: IF (12 X AVG. MPO) IS LESS THAN MIN. QUANTITY THEN USE MIN. QTY. X PRICE EA. + MASK CHARGE = 12 MO. COST.)

(COST OF EPROM)

12	X	AVG. MPO	X	PRICE EA.	=	12 MO. COST OF EPROM
1 YR.						

COST SAVINGS FOR NEXT 12 MONTHS MPO: $_____.

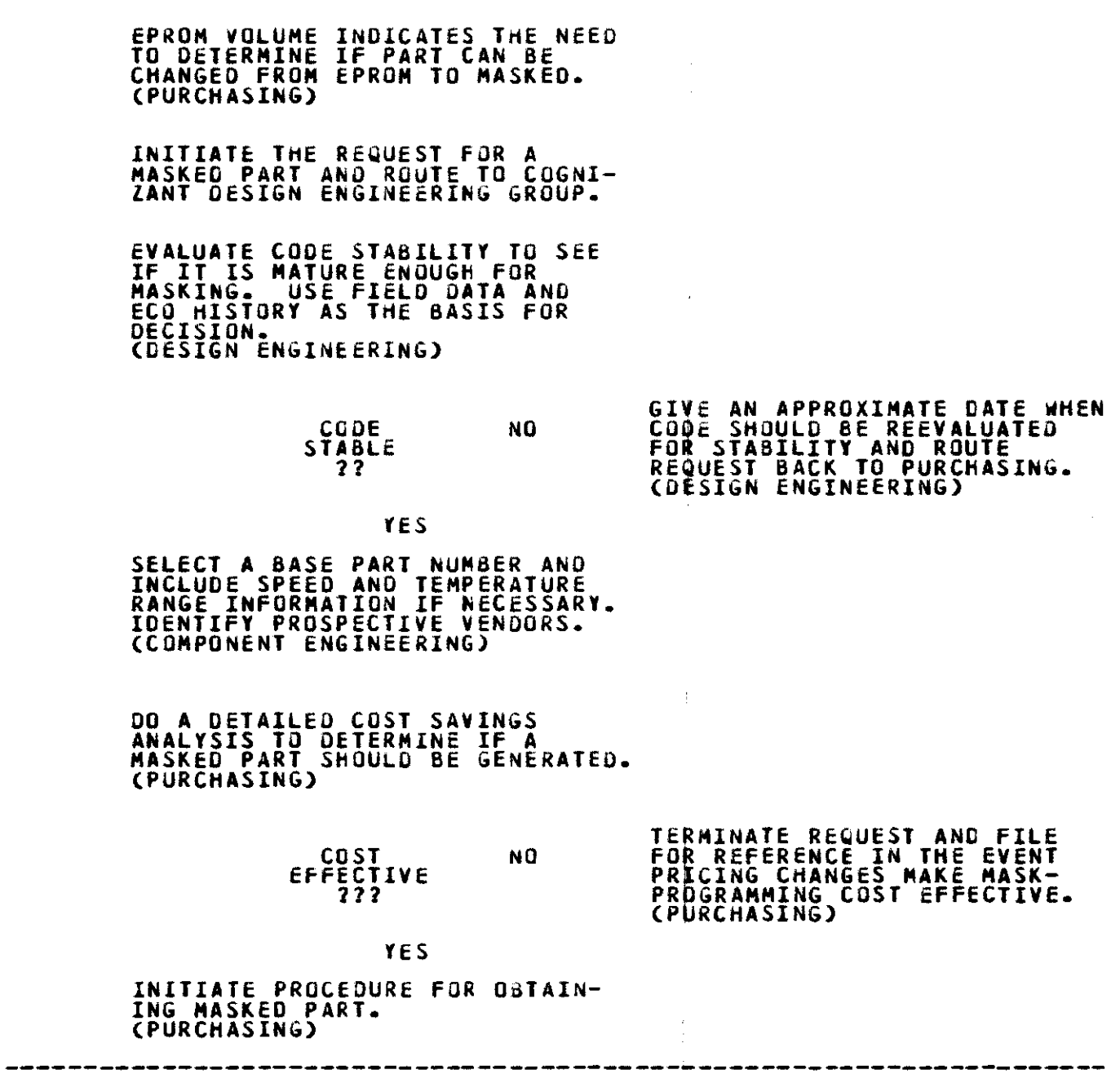
V. DECISION (PURCHASING)

* YES, SAVINGS JUSTIFY THE GENERATION OF A MASKED PART. INITIATE PROCEDURE FOR OBTAINING MASKED PART.
NO, MASKING IS NOT YET COST EFFECTIVE. RE-EVALUATE ON:_____
(DATE)

AUTHORIZATION: _____ DATE: _____

*(IF YES, FORWARD COPIES OF THIS FORM TO THE COMPONENT ENGINEER AND THE DESIGN ENGINEER.)

FIGURE 25.1 FLOWCHART OF PROCEDURE FOR CHANGING EPROM TO MASKED-PROGRAMMED PART



SOFTWARE RELEASE AND DOCUMENTATION PROCEDURES

***NOTE: 6 JULY 1983, S. RUSSELL
THIS CHAPTER NEEDS COMPLETELY REWRITTEN TO CONFORM TO THE NEW PROCEDURES AND TO CONSOLIDATE MATERIAL WRITTEN BY DEBBIE AND I.

** 9.0 SOFTWARE RELEASE AND DOCUMENTATION PROCEDURES

** 9.1 INTRODUCTION

NEW REQUIREMENTS FOR SOFTWARE CONFIGURATION MANAGEMENT SUCH AS DO-178 AND THE RECENT INCREASES IN THE SIZE AND COMPLEXITY OF SOFTWARE PRODUCED AT KING RADIO ARE CREATING THE NEED TO HAVE MORE FORMAL SOFTWARE RELEASE AND DOCUMENTATION METHODS. THIS CHAPTER DESCRIBES THOSE RELEASE AND DOCUMENTATION PROCEDURES. IT IS INTENDED TO GIVE AN OVERVIEW OF THE STEPS INVOLVED DURING THE DESIGN AND DEVELOPMENT PHASE OF A PROJECT. DETAILED DESCRIPTIONS OF EACH SOFTWARE CONTROL FORM AND THE CONTENTS OF THE VARIOUS DOCUMENTS ARE GIVEN IN OTHER CHAPTERS.

THE RELEASE STRATEGY HAS BEEN DESIGNED TO ACCOMPLISH THE FOLLOWING:

1) SATISFY THE REQUIREMENTS OF GOOD CONFIGURATION MANAGEMENT PRACTICES.
2) INTEGRATE THE PREPARATION, REVIEW, AND RELEASE OF DOCUMENTATION WITH THE DESIGN AND DEVELOPMENT PROCESS.
3) TIME THE RELEASE OF DOCUMENTS SO THEY WILL HAVE MAXIMUM USEFULNESS IN CARRYING OUT SUBSEQUENT DESIGN AND DEVELOPMENT.
4) PROVIDE FOR DOCUMENT REVIEW CONCURRENT WITH DESIGN REVIEWS.
5) DISTRIBUTE THE DOCUMENT PREPARATION EFFORT TO AVOID 'PEAKING'.

WHEN RELEASES ARE PLANNED IN STAGES, PROJECT MANAGEMENT CAN HAVE CONTINUING VISIBILITY INTO PROJECT STATUS AND, AT THE SAME TIME, THE DOCUMENTATION EFFORT CAN BE SPREAD OVER THE SAME PERIOD AS THE DESIGN AND DEVELOPMENT. AT EACH STAGE, DOCUMENTS HAVE BEEN SELECTED FOR RELEASE THAT WILL BE USEFUL IN CARRYING OUT SUBSEQUENT DESIGN AND DEVELOPMENT.

IF DOCUMENTATION IS NOT CARRIED ALONG WITH DESIGN AND DEVELOPMENT, THE DESIGN TEAM WILL BE OVERLOADED WITH PREPARATION TASKS DURING THE CRITICAL LAST PHASES OF THE PROJECT. WITHOUT TIMELY DOCUMENTATION, THE DESIGN REVIEWS CANNOT BE ADEQUATELY SUPPORTED AND THE DOCUMENTATATION DOES NOT RECEIVE A REVIEW. THIS INCREASES THE DANGER THAT INADEQUATELY REVIEWED AND DOCUMENTED SOFTWARE WILL BE RELEASED INTO PRODUCTS.

IT IS RECOMMENDED THAT A MAJOR DESIGN REVIEW BE CONDUCTED PRIOR TO EACH SOFTWARE RELEASE. THE REVIEW SHOULD BE SCHEDULED SO AS TO ALLOW TIME FOR MINOR CHANGES TO BE MADE TO DOCUMENTATION PRIOR TO THE RELEASE DEADLINE. EACH DOCUMENT SHOULD BE REVIEWED FOR:

1) COMPLETENESS AND ACCURACY.
2) CONFORMANCE TO COMPANY STANDARDS FOR GOOD ENGINEERING PRACTICES.
3) COMPLIANCE WITH THE REQUIREMENTS OF DO-178.

THE FINAL RELEASE OF LOADABLE CODE IS SCHEDULED AS CLOSE TO THE END OF THE DESIGN AND DEVELOPMENT CYCLE AS POSSIBLE. THIS IS DESIRABLE BECAUSE IT AVOIDS THE DOCUMENTATION BURDEN OF REVISION CONTROL DURING THE HIGHLY FLUID DEVELOPMENT PHASE WITHOUT COMPROMISING DESIRABLE CONFIGURATION MANAGEMENT GOALS.

WE WILL NOW DESCRIBE PROJECT STARTUP AND THE MAJOR PROJECT RELEASE AND DOCUMENTATION MILESTONES. BELOW IS A TABLE OF DOCUMENT NUMBERS THAT ARE ASSIGNED TO SOFTWARE. APPLICATIONS FOR THESE NUMBERS ARE MADE ON SPECIAL FORMS AVAILABLE FROM JAN O'DELL. THESE NUMBERS WILL BE ENTERED INTO THE ENGINEERING MASTER DATA STRUCTURE ON VAX. FIGURE 9-1 SHOWS THE COMPLETE STRUCTURE OF SOFTWARE DOCUMENTS STARTING AT THE SYSTEM LEVEL AND GOING THROUGH THE LRU AND PROCESSOR LEVELS TO THE PROGRAMMED DEVICE.

TABLE OF SOFTWARE DOCUMENT NUMBERS AND NAMES

DOCUMENT NO.	NAME
700-XXXX-RN	SYSTEM CID
701-XXXX-RN	SYSTEM SPECIFICATION
705-XXXX-RN	SOFTWARE REQUIREMENTS
706-XXXX-RN	SOFTWARE MANAGEMENT PLAN
707-XXXX-RN	LRU SOFTWARE TEST
708-XXXX-RN	SOFTWARE EXECUTIVE SUMMARY
715-XXXX-RN	LRU SOFTWARE
716-XXXX-RN	LRU SOFTWARE STRUCTURE DIAGRAM
718-XXXX-RN	PROGRAMMER'S REFERENCE LITERATURE
719-XXXX-RN	SOURCE LISTING
720-XXXX-RN	LOAD FILE RELEASE
721-XXXX-RN	DESIGN DESCRIPTION
722-XXXX-RN	BINARY IMAGE FILE

** 9.2 SYSTEM DEFINITION AND DESIGN.

THE FIRST TASK IN A TOP-DOWN APPROACH TO NEW PRODUCTS IS THE PREPARATION OF A DETAILED PRODUCT DEFINITION. THIS DEFINITION USUALLY INCLUDES OPERATIONAL CHARACTERISTICS, CONTROL AND DISPLAY, MAJOR ELECTRICAL AND ENVIRONMENTAL SPECIFICATIONS, LRU SYSTEM CONFIGURATION, AND TARGET GOALS FOR SIZE, WEIGHT, POWER, AND COST.

AT PROJECT STARTUP, THE PROJECT ENGINEERING MANAGER (USUALLY, THOUGH NOT ALWAYS, THE GROUP LEADER) AND HIS STAFF MUST WORK WITH TOP MANAGEMENT, OR AN OUTSIDE CUSTOMER, TO PRODUCE THE DETAILED PRODUCT DEFINITION. SOMETIMES A VERY DETAILED PRODUCT DEFINITION IS PROVIDED BUT FREQUENTLY ONLY A GENERAL CONCEPT OF THE PRODUCT AND SOME OF ITS DEFINITIONS ARE AVAILABLE. THE PRODUCT TEAM SHOULD GENERATE PRELIMINARY PRODUCT DEFINITIONS AND REVIEW THEM WITH TOP MANAGEMENT, OR THE OUTSIDE CUSTOMER, UNTIL THE FIRST BASELINE DEFINITION IS SATISFACTORY TO ALL CONCERNED. IT NORMALLY TAKES A FEW ITERATIONS TO PRODUCE ENOUGH DETAILED INFORMATION TO PROCEED WITH THE GENERATION OF THE FIRST SYSTEM ENGINEERING DOCUMENTS, THE SYSTEM SPECIFICATION FOR THE PRODUCT AND THE SYSTEM CONFIGURATION INDEX (CID).

AFTER THE SYSTEM SPECIFICATION DOCUMENT HAS BEEN APPROVED BY TOP MANAGEMENT, OR THE OUTSIDE CUSTOMER, THE FUNCTIONAL REQUIREMENTS ARE PARTITIONED INTO HARDWARE AND SOFTWARE TASKS. WHEN THE SOFTWARE TASKS ARE FULLY IDENTIFIED AND DEFINED, WORK STARTS ON THE SOFTWARE REQUIREMENTS DOCUMENT.

IS THIS WHERE THE STATEMENT OF WORK COMES IN????????

AT THIS TIME, THE PROJECT ENGINEERING MANAGER (GROUP LEADER) WILL HAVE ENOUGH INSIGHT INTO THE REQUIREMENTS OF THE PROJECT, AND ITS LEVEL OF EFFORT, THAT HE CAN ORGANIZE A SOFTWARE MANAGEMENT PLAN AND PREPARE THE SOFTWARE MANAGEMENT PLAN DOCUMENT.

DESIGN WORK STARTS.

9.3 TOP-LEVEL SOFTWARE RELEASE (STAGE-1).

WHEN THE DESIGN AND DEVELOPMENT PHASE IS ABOUT ONE-THIRD COMPLETE, A MAJOR SOFTWARE MILESTONE SHOULD BE THE TOP-LEVEL SOFTWARE (STAGE-1) RELEASE. CONSIDERABLE DESIGN WORK HAS BEEN COMPLETED AND THE PROJECT TEAM IS NOW ABLE TO FINALIZE SOFTWARE REQUIREMENTS. A PROJECT LEVEL DESIGN REVIEW SHOULD BE HELD BY THE PROJECT ENGINEERING MANAGER (GROUP LEADER) TO CONSIDER DESIGN STATUS AND DESIGN ISSUES. THE DOCUMENTATION TO BE RELEASED SHOULD ALSO BE REVIEWED FOR ACCURACY AND COMPLETENESS. AFTER THE REVIEW, THE DOCUMENTS SHOULD BE CORRECTED AS NECESSARY AND A STAGE-1 RELEASE FORM (799-0001-00) COMPLETED AND SUBMITTED TO SOFTWARE CONFIGURATION MANAGEMENT. THE DOCUMENTS RELEASED AT THIS TIME ARE THE SOFTWARE REQUIREMENTS DOCUMENT AND THE SOFTWARE MANAGEMENT PLAN DOCUMENT. PART OF THE RELEASE PROCEDURE IS THE PREASSIGNMENT OF NUMBERS FOR SOFTWARE DOCUMENTS TO BE RELEASED AT THE NEXT MILESTONE.

9.4 LRU SOFTWARE DOCUMENTATION RELEASE (STAGE-2).

WHEN THE DESIGN AND DEVELOPMENT PHASE IS ABOUT TWO-THIRDS COMPLETE, OR WHEN MOST OF THE DESIGN ISSUES HAVE BEEN DECIDED AND A MAJOR PART OF THE BASELINE DEVELOPMENT HAS BEEN COMPLETED, THE LRU (UNIT) SOFTWARE DOCUMENTATION SHOULD BE RELEASED.

JUST BEFORE RELEASE, ANOTHER MAJOR PROJECT-LEVEL DESIGN REVIEW SHOULD BE HELD TO CLARIFY REMAINING DESIGN ISSUES AND TO REVIEW THE DOCUMENTATION ABOUT TO BE RELEASED FOR ACCURACY AND COMPLETENESS. AFTER ANY NEEDED CORRECTIONS ARE MADE, A STAGE-2 RELEASE FORM IS COMPLETED FOR EACH LRU IN THE SYSTEM AND SUBMITTED TO SOFTWARE CONFIGURATION MANAGEMENT.

THIS RELEASE PROVIDES ALL OF THE DOCUMENTS THAT DESCRIBE PRODUCT SOFTWARE AT THE LRU LEVEL. THE LRU SOFTWARE DOCUMENT (715-XXXX-RN) IS THE MAIN SOFTWARE REFERENCE FOR ALL THE SOFTWARE CONTAINED IN THE LRU. ALSO INCLUDED ARE THE LRU SOFTWARE TEST DOCUMENT, AND THE LRU SOFTWARE STRUCTURE DIAGRAM.

9.5 SPECIFICATION CONTROL DRAWING RELEASES

AFTER THE STAGE-2 RELEASE IS COMPLETED BUT BEFORE THE PURCHASING BILL-OF-MATERIALS IS RELEASED, AN APPLICATION IS MADE TO COMPONENT ENGINEERING FOR THE RELEASE OF A SPECIFICATION CONTROL DRAWING FOR EACH PROGRAMMED PROCESSOR OR MEMORY CHIP IN EACH NEW LRU IN THE SYSTEM. THE PRINCIPAL REASON FOR DOING THIS IS TO OBTAIN A PART NUMBER THAT CAN BE PUT ON THE BILL-OF-MATERIALS. THIS PART NUMBER AND ITS ASSOCIATED SPECIFICATION CONTROL DRAWING WILL BE USED BY PURCHASING PLANNERS TO SCHEDULE THE DELIVERY OF ELECTICAL PARTS. THIS RELEASE REQUIRES THREE STEPS:

1. APPLY TO COMPONENT ENGINEERING FOR A 122-XXXX-RN NUMBER ASSIGNMENT. THIS DOES NOT RELEASE THE PRINT BUT PROVIDES A RESERVED PART NUMBER TO BE USED ON THE BILL-OF-MATERIALS.
2. APPLY TO DOCUMENT CONTROL FOR A 722-XXXX-RN DOCUMENT NUMBER FOR THE BINARY IMAGE FILE THAT DESCRIBES THE PROGRAM IN THE CHIP.
3. WHEN THE BINARY IMAGE FILE DOCUMENT NUMBER HAS BEEN ASSIGNED, APPLY TO COMPONENT ENGINEERING REQUESTING RELEASE OF THE 122-XXXX-RN SPECIFICATION CONTROL DRAWING THAT APPLIES TO THAT PROGRAMMED CHIP.

AT THIS POINT, THE PROGRAMMED I.C. SPECIFICATION CONTROL DRAWING WILL BE AVAILABLE TO THE PURCHASING PLANNERS SO THEY CAN SPECIFY AND OBTAIN THE TARGER ELECTRICAL PART AND DETERMINE IF THE PART WILL EVER BE MASKED. SINCE THE SOFTWARE HAS NOT YET BEEN RELEASED, NO MASTER PROGRAMMED CHIPS ARE AVAILABLE.

** 9.6 LOAD FILE SOFTWARE RELEASE (STAGE-3).

THE LAST MAJOR SOFTWARE RELEASE OCCURS AT THE END OF DESIGN AND DEVELOPMENT. AT THIS TIME, THE SOFTWARE HAS BEEN VERIFIED AND VALIDATED AND IS MATURE ENOUGH TO GO INTO THE MAINTAINANCE STAGE OF ITS LIFE CYCLE. THIS WILL USUALLY OCCUR JUST BEFORE MANUFACTURING RELEASE OF THE ENTIRE PRODUCT. THIS RELEASE IS CALLED THE LOAD FILE RELEASE BECAUSE IT INITIATES THE PROCESS OF CREATING THE FIRST VERSION OF LOADABLE SOFTWARE THAT IS PUT UNDER REVISION CONTROL.

A MAJOR GOAL OF THE LOAD FILE RELEASE IS TO PROVIDE COMPLETE DOCUMENTATION OF ALL TARGET SOFTWARE AND SUPPORT SOFTWARE AND PROVIDE THE NECESSARY INFORMATION TO RECREATE THE LOAD FILE AND ITS VARIOUS FUTURE REVISIONS AT ANY TIME DURING THE LIFE CYCLE OF THE PRODUCT. IT ALSO CREATES THE ASCII-HEX FILES NEEDED FOR DOWNLOAD AND INITIATES THE PROCESS OF CREATING THE MASTER CHIPS.

TO BEGIN, THE PROJECT ENGINEER COMPLETES A STAGE-3 RELEASE FORM FOR EACH PROCESSOR IN THE NEW SYSTEM. INFORMATION REQUIRED INCLUDES ASSIGNED DOCUMENT NUMBERS, SUPPORT SOFTWARE COMMAND PROCEDURE FILE NAME, TARGET SOURCE CODE FILE NAMES, AND THE EXPLICIT FILE NAMES FOR ALL SUPPORT SOFTWARE. THE PROJECT ENGINEER ALSO PROVIDES THE SOFTWARE CONTROL LIBRARIAN WITH FOUR ERASED UV PROM CHIP TO BE USED AS MASTER COPIES.
8888888888888888888888888888888888888888888888888888888888888888888888888888
COMBINE THE FOLLOWING THREE PARAGRAPHS
8888888888888888888888888888888888888888888888888888888888888888888888888888
THE NEXT SEVERAL LINES CONTAIN DOCUMENTATION REFERENCES AT THE PROGRAMMED MEMORY DEVICE LEVEL. A GENERIC CLASSIFICATION CALLED BINARY IMAGE IS USED BECAUSE THESE FILES REPRESENT VARIOUS FORMATS OF THE ACTUAL BINARY CODE IN THE PROGRAMMED DEVICE.

THE FIRST TYPES OF FILES ARE THE EVEN/ODD BYTE FILES. THESE ARE NEEDED WHEN THE LOAD FILE FOR A 16-BIT PROCESSOR MUST BE LOADED INTO BYTE-ORGANIZED MEMORY. A SUPPORT SOFTWARE PROGRAM CALLED THE SEPARATOR IS USED TO SORT THE ODD AND EVEN BYTES AND SEPARATE THEM INTO TWO FILES HAVING THE SAME FILE NAME AS THE LOAD FILE BUT DIFFERENTIATED BY UNIQUE FILE EXTENSIONS. FOR 8-BIT PROCESSORS, THIS BLANK IS LABELED N/U BECAUSE THIS STEP IS NOT USED. THESE FILES ARE SAVED ON VAX AND ARCHIVED UNDER THE LOAD FILE SAVE-SET NAME BUT ARE NOT PUT INTO A RELEASED DOCUMENT.

THE NEXT STEP IS TO CREATE A BINARY IMAGE DOCUMENT FILE THAT IS A LISTING OF ALL MEMORY LOCATIONS AND CORRESPONDING OP CODES IN THE TARGET MEMORY DEVICE. ONE OF THESE FILES FOR EACH MEMORY DEVICE WILL BE RELEASED AS A HARD-COPY DOCUMENT OF THE CODE IN THE CHIP.

THE NEXT LINES CONTAIN DOCUMENTATION REFERENCES AT THE PROGRAMMED MEMORY DEVICE LEVEL. FOR MOST FUTURE PROJECTS, THE LOAD FILE CREATED BY THE LINKING LOADER WILL BE LARGER THAN CAN BE PROGRAMMED INTO A SINGLE CHIP. ALSO, THE LOAD FILE FOR 16-BIT PROCESSORS WILL, MOST LIKELY, HAVE TO BE LOADED INTO BYTE-ORGANIZED MEMORY. TO TAKE CARE OF THESE FACTORS, MORE SUPPORT SOFTWARE IS AVAILABLE TO SEPARATE THE SINGLE LOAD FILE INTO ODD AND EVEN BYTES, PARTITION THE LOADABLE OBJECT CODE INTO EACH PHYSICAL MEMORY DEVICE, AND FORMAT AN ASCII-HEX FILE FOR EACH MEMORY DEVICE SO ITS CORRESPONDING CODE CAN BE DOWNLOADED TO A PROM PROGRAMMER OR INTEGRATION UNIT.

THE CODE IN EACH MEMORY DEVICE WILL BE DOCUMENTED BY A BINARY IMAGE FILE. THIS DOCUMENT IS A LISTING OF THE MEMORY LOCATIONS AND OP CODE APPLICABLE TO EACH PROGRAMMED DEVICE. EACH BINARY IMAGE FILE HAS A 722-XXXX-RN DOCUMENT NUMBER AND A DOCUMENT FILE NAME ON VAX. THIS DOCUMENT IS A PRINTOUT OF THE CODE IN THE DEVICE. ANOTHER SUPPORT SOFTWARE PACKAGE USES THIS DOCUMENTED CODE TO CREATE THE ACTUAL ASCII-HEX FILE THAT WILL BE USED TO PROGRAM EACH DEVICE. THE ASCII-HEX FILE NAME WILL BE THE NINE DIGIT DOCUMENT NUMBER FOR THE BINARY IMAGE FILE WITH A FILE EXTENSION THAT CORRESPONDS TO THE DESIRED HEX FORMAT.

THE BINARY IMAGE FILE DOCUMENT NUMBER (722-XXXX-RN) IS ASSIGNED AT THIS TIME. BOTH THE DOCUMENT AND ASCII-HEX FILES ARE GIVEN NAMES CORRESPONDING TO TO THE SEVEN-DIGIT BINARY IMAGE FILE DOCUMENT NUMBER. THE FILE EXTENTION ON THE ASCII-HEX FILE IS SPECIFIC TO DESIRED HEX FORMAT (.HEX FOR INTEL, .LX FOR MOTOROLA, .TEK FOR TEKHEX). THE CORRESPONDING PROGRAMMED I.C. PART NUMBER IS REQUIRED SO THAT THE MASTER CHIPS CAN BE CORRECTLY LABELED.

SINCE THE LOAD FILE IS ALSO AVAILABLE IN ASCII-HEX FORMAT, IT MAY CAUSE SOME CONFUSION AS TO WHY THE ADDITIONAL FILES ARE CREATED. THE BINARY IMAGE FILE IS USED AS A HARD-COPY DOCUMENTATION OF THE CODE IN EACH MEMORY DEVICE. AS AN ADDITIONAL CHECK OF THE ACCURACY OF THIS DOCUMENT, THE BINARY IMAGE FILE IS ACTUALLY USED TO RE-CREATE A DOWNLOADABLE ASCII-HEX FILE. IT IS THIS FILE THAT IS USED TO PROGRAM THE MASTER DEVICES THAT WILL BE USED IN PRODUCTION.
88888888888888888888888888888888888888888888888888888888888888888888888888

AFTER THE FORM IS COMPLETED AND SUBMITTED TO SOFTWARE CONTROL, THE COMMAND PROCEDURES AND THE SOURCE FILES ARE COPIED TO THE [EXCHANGE] DIRECTORY SO THEY ARE AVAILABLE TO SOFTWARE CONTROL. THE SOFTWARE CONTROL LIBRARIAN WILL COPY THESE FILES INTO A SUBDIRECTORY UNDER THE SOFTWARE CONTROL DIRECTORY. THE SUBDIRECTORY WILL HAVE A 9-DIGIT NAME [*.720XXXXRN] CORRESPONDING TO THE DOCUMENT NUMBER (720-XXXX-RN) OF THE LOAD FILE.

THE LIBRARIAN THEN EXECUTES THE COMMAND FILE TO RUN ALL OF THE SUPPORT SOFTWARE REQUIRED TO CREATE THE LOAD FILE, ODD AND EVEN BYTE FILES (IF NEEDED), BINARY IMAGE DOCUMENT FILES, AND THE DOWNLOADABLE ASCII-HEX FILES FOR EACH MEMORY DEVICE. ALL OF THESE FILES WILL BE IN THE [*.720XXXXRN] SUBDIRECTORY.

TO COMPLETE THE STAGE-3 RELEASE, THE LIBRARIAN WILL CREATE FOUR MASTER COPIES OF EACH UV MEMORY DEVICE USING THE ASCII-HEX FILES. ONE OF THESE MASTERS WILL BE GIVEN TO THE PROJECT ENGINEER SO ITS CODE MAY BE VERIFIED AGAINST THE CORRESPONDING DEVICE USED IN THE TESTED ENGINEERING MODEL. WHEN THE VERIFICATION PROCESS IS COMPLETED, THE PROJECT ENGINEER WILL SIGN THE STAGE-3 RELEASE FORM TESTIFYING THAT THE CODE IS IDENTICAL TO THAT IN THE ENGINEERING MODEL. AT THIS TIME, THE LIBRARIAN WILL SUBMIT THE THREE REMAINING MASTER COPIES OF THE PROGRAMMED DEVICE TO COMPONENT ENGINEERING. TWO OF THESE MASTERS WILL THEN BE AVAILABLE FOR USE BY RECEIVING INSPECTION TO GANG PROGRAM PRODUCTION PARTS.

TO COMPLETE THE STAGE-3 RELEASE, THE SOFTWARE CONTROL LIBRARIAN WILL ARCHIVE ALL OF THE FILES IN THE [*.720XXXXRN] SUBDIRECTORY ONTO MAGNETIC TAPE IN A CONTAINER FILE (SAVE_SET) WITH THE NAME 720XXXXRN. ALSO THE LOAD FILE DOCUMENT, THE BINARY IMAGE FILE DOCUMENT, AND THE SOURCE FILE DOCUMENT WILL BE PREPARED AND RELEASED INTO THE SYSTEM. THE STAGE-3 RELEASE IS NOW COMPLETE.

SOFTWARE RELEASE PROCEDURE FOR BEGINNERS

* PLEASE NOTE THAT THIS PROCEDURE IS INFORMAL AND UNOFFICIAL; FURTHER, IT DOES NOT DEAL WILL ALL ASPECTS OF SOFTWARE CONFIGURATION MANAGEMENT. THIS DOCUMENT IS INTENDED ONLY TO PROVIDE SUFFICIENT INFORMATION TO ALLOW THE RELEASE OF NEW SOFTWARE INTO THE SYSTEM UNDER THE NEW FORMAT. IT DOES NOT GO INTO DETAIL ON THE REQUIREMENTS FOR SOME OF THE OTHER NECESSARY DOCUMENTATION; IT MERELY REQUIRES THAT THE NUMBERS FOR THOSE DOCUMENTS BE ASSIGNED. PLEASE CONSULT WITH SOFTWARE CONTROL AS TO YOUR OBLIGATIONS WITH RESPECT TO GENERATING 715-, 719-, AND 721- DOCUMENTS. *

INTRODUCTION

SOFTWARE RELEASES MAY BE EITHER NEW RELEASES OR REVISIONS TO EXISTING SOFTWARE. FOR THE NEW RELEASE, TWO FORMS ARE USED: THE 'NEW SOFTWARE: PRELIMINARY RELEASE' , AND THE 'LOAD FILE RELEASE ORDER'. FOR REVISIONS TO EXISTING SOFTWARE, THE LOAD FILE RELEASE ORDER ONLY IS USED. PLEASE PROCEED ACCORDING TO SECTION A FOR NEW RELEASES; ACCORDING TO SECTION B FOR REVISIONS TO EXISTING SOFTWARE.

RELEASES OF EXISTING SOFTWARE WHICH HAS REMAINED IN THE SYSTEM WITH A 120- AS OPPOSED TO A 122- NUMBER MUST BE EFFECTED ACCORDING TO SECTION A.

IT IS UNDERSTOOD THAT A SOFTWARE RELEASE IS THE RELEASE OF A PROCESSOR SYSTEM. A SINGLE SYSTEM MAY INVOLVE MORE THAN ONE PHYSICAL PROGRAMMABLE COMPONENT AND MAY THUS INCORPORATE MORE THAN ONE 122- NUMBER; HOWEVER, IF A MULTIPLE SYSTEM IS CONTROLLED BY SEVERAL MICROCOMPUTERS, EACH MUST BE RELEASED SEPARATELY AND HAVE ITS OWN PAPERWORK.

A: NEW RELEASES

NEW RELEASES (FOR 122- NUMBERS ENDING IN -00) WILL BE EFFECTED IN TWO STAGES.

THE FIRST STAGE OCCURS AT THE SAME TIME AS THE PURCHASING (INCLUDING THE BILL OF MATERIAL) RELEASE. AT THIS TIME APPROPRIATE PART NUMBERS WILL BE ASSIGNED AND A 122- NUMBER PIECE PARTS PRINT WILL BE GENERATED.

THE SECOND STAGE OCCURS AT THE SAME TIME AS THE MANUFACTURING RELEASE. AT THIS TIME ALL THE INFORMATION PERTAINING TO THE PROCESSOR SYSTEM WILL BE GATHERED IN ONE DOCUMENT AND SUBMITTED TO SOFTWARE CONTROL. SOFTWARE CONTROL WILL CREATE MASTER CHIPS USING A DOWNLOADABLE FILE DERIVED FROM THE SOFTWARE ENGINEER'S SOURCE FILE. THESE MASTER CHIPS WILL BE VERIFIED AND VALIDATED BY THE SOFTWARE PROJECT ENGINEER, AND WILL THEN BE DELIVERED TO COMPONENT ENGINEERING. COMPONENT ENGINEERING WILL KEEP ONE MASTER CHIP AS A RECORD; THE REMAINING TWO CHIPS WILL BE DELIVERED TO RECEIVING INSPECTION, WHERE THEY WILL BE USED FOR THE PROGRAMMING OF MICROPROCESSOR CHIPS FOR PRODUCTION.

STAGE 1: NEW SOFTWARE: PRELIMINARY RELEASE

IN THIS STAGE, THE SOFTWARE PROJECT ENGINEER MUST OBTAIN KING PART NUMBERS FOR THE PROCESSOR AND FOR EACH PROGRAMMED MICROPROCESSOR IN HIS PROCESSOR SYSTEM. THE PROCESSOR SYSTEM NUMBER WILL ALWAYS BEGIN WITH 720-; THE PROGRAMMED DEVICE NUMBER WILL ALWAYS BEGIN WITH 122- .
THE ENGINEER MUST ALSO OBTAIN A PART NUMBER FOR THE BINARY IMAGE FILE ASSOCIATED WITH EACH 122- NUMBER. THE BINARY IMAGE FILE IS A PRINTED ASCII-HE REPRESENTATION OF THE CONTENTS OF THE PROGRAMMED DEVICE. THE BINARY IMAGE FIL NUMBER WILL ALWAYS BEGIN WITH 722- .

OBTAIN THE DOCUMENT ENTITLED 'NEW SOFTWARE: PRELIMINARY RELEASE' FROM SOFTWARE CONTROL. FILL IT IN AS FOLLOWS:

1. FILL IN THE DATE ON WHICH YOU ARE MAKING THE REQUEST. IF YOU DON'T COMPLETE THE PRELIMINARY RELEASE PROCEDURE FOR SEVERAL WEEKS AFTER FILLING IN THE DATE, PLEASE CHANGE THE DATE.

2. IF YOU ARE THE SOFTWARE PROJECT ENGINEER, FILL IN YOUR NAME AND EXTENSION. IF YOU ARE NOT THE SOFTWARE PROJECT ENGINEER, YOU SHOULD NOT BE FILLING IN THIS FORM.

3. FILL IN THE UNIT NUMBER AND NAME OF EACH UNIT THE PROCESSOR WILL BE USED IN. DEFINE 'UNIT' AS A RADIO, A BOX, A DISCRETE PIECE OF EQUIPMENT WHICH HAS ITS OWN FINAL COVERS. FOR INSTANCE, 'KKK 191 AUTOPILOT' AND 'KKK 192 AUTOPILOT' MIGHT BE ENTERED ON TWO LINES HERE. IF APPROPRIATE, HOWEVER, YOU MAY ENTER TWO UNIT NAMES ON ONE LINE, AS 'KY 196/197 VHF COMM'.

4. FILL IN THE NAME OF THE PROCESSOR SYSTEM. THIS MEANS THAT YOU NEED TO DECIDE WHAT IT IS GOING TO BE IF YOU HAVE NOT DONE SO ALREADY. TRY TO MAKE YOUR SYSTEM NAME UNIQUE, AND DESCRIPTIVE ENOUGH SO THAT A PERSON NOT INTIMATELY ASSOCIATED WITH THE PROJECT COULD GET SOME IDEA OF WHAT THE PROCESSOR CONTROLS. TRY TO INCLUDE A UNIT OR A HARDWARE SYSTEM NAME IN THE PROCESSOR NAME. FOR INSTANCE, '192 AP PITCH AXIS' GIVES AN IDEA OF WHERE THE PROCESSOR IS USED AND OF WHAT IT DOES. IN CASES WHERE THERE IS ONLY ONE PROCESSOR IN A UNIT AND IT IS THE CONTROL PROCESSOR FOR ALL FUNCTIONS, A TITLE SUCH AS '196/7 VHF COMM CONTROLLER' IS BETTER THAN 'CONTROLLER'.

5. FILL IN THE PROCESSOR NUMBER. IT WILL ALWAYS START WITH 720-. IT WAS ISSUED TOGETHER WITH THE 719- AND 721- NUMBERS ON THE 'INDIVIDUAL PROCESSOR PART NUMBERS' FORM WHICH YOU SHOULD HAVE IN YOUR FILES.

6. FOR EACH PROGRAMMED DEVICE IN YOUR PROCESSOR SYSTEM, FILL IN ONE OF THE SECTIONS BETWEEN ASTERISK LINES. THERE IS ROOM ON THE FORM FOR THREE DEVICES IN ONE SYSTEM. IF YOUR PROCESSOR SYSTEM HAS MORE, ATTACH ANOTHER SHEET TO THE FIRST WITH THE INFORMATION FOR THE ADDITIONAL DEVICES. YOU NEED NOT FILL IN THE TOP SECTION (NUMBERS 1 THROUGH 5) AGAIN. FILL IN THE SECTION BETWEEN ASTERISKS AS FOLLOWS:

7. DECIDE WHAT YOU WANT THE STANDARD DESCRIPTION FOR THE DEVICE TO BE. YOU MAY USE UP TO 18 CHARACTERS. THE STANDARD DESCRIPTION IS THE DESCRIPTION THAT WILL BE PUT ON VAX AND THAT WILL BE PRINTED ON THE BILL OF MATERIALS. TRY TO INCLUDE SOME INFORMATION ABOUT THE REVISION OF THE SOFTWARE, THE FUNCTION OF THE DEVICE, AND WHETHER IT IS ELECTRICALLY PROGRAMMED OR MASKED. FOR INSTANCE IN A TWO-DEVICE SYSTEM FOR A KXX 777, YOU MIGHT HAVE STANDARD DESCRIPTIONS OF '777 SYNTH A V00E' AND '777 SYNTH B V00E', WHERE THE 777 WAS INFORMATON ABOUT THE UNIT THE DEVICE WAS USED IN; SYNTH WAS INFORMATION ABOUT THE DEVICE'S FUNCTION; A OR B WAS INFORMATION SAYING THAT THERE WAS MORE THAN ONE SUCH DEVICE, AND WHICH ONE THIS DEVICE WAS; V00 WAS INFORMATION ABOUT THE REVISION LEVEL OF THE DEVICE; AND E WAS INFORMATION ABOUT THE TYPE OF PROGRAMMING. THI IS NOT A REQUIREMENT (IT IS RECOGNIZED THAT 18 CHARACTERS IS NOT AN ENORMOUS QUANTITY) BUT IT DOES MAKE IT MUCH EASIER FOR PERSONS NOT INTIMATELY ASSOCIATE WITH THE PROJECT TO DEAL WITH THE STANDARD DESCRIPTIONS. THE STANDARD DESCRIPTION MAY BE THE SAME AS FOR THE 720- NUMBER, BUT REMEMBER THAT SINCE YO NEED TO BE ABLE TO DISTINGUISH YOUR DOCUMENTS FROM EACH OTHER, IT MAY BE A GOO IDEA NOT TO GIVE THEM ALL THE SAME DESCRIPTION. FILL IN THE STANDARD DESCRIPTION YOU HAVE DECIDED ON IN THE BOXES TO THE RIGHT OF THE SPACE FOR THE 122- NUMBER.

8. FILL IN THE NUMBER OF THE UNPROGRAMMED DEVICE YOU WILL BE USING TO CREATE YOUR PROGRAMMED PARTS. THIS MEANS THAT YOU NEED TO HAVE MADE A DECISION ABOUT WHICH UNPROGRAMMED PART YOU ARE GOING TO USE. DO NOT FILL IN A NUMBER THAT YOU 'THINK' YOU 'MIGHT' USE JUST TO GET IT OUT OF THE WAY; PURCHASING IS GOING TO USE THIS INFORMATION TO PREPARE FOR THE MANUFACTURING RELEASE OF YOUR UNIT OR SYSTEM, AND IT IS USELESS FOR THEM TO PROCEED ON ERRONEOUS OR INDEFINITE INFORMATION. IF YOU REALLY DON'T KNOW YET WHAT PART YOU ARE GOING TO USE, YOU ARE NOT READY TO RELEASE THE BILL OF MATERIALS OR TO DO A PRELIMINARY RELEASE OF THE SOFTWARE.

IF THE UNPROGRAMMED DEVICE IS ONE WHICH HAS NOT PREVIOUSLY BEEN USED ON A KING UNIT, YOU NEED TO HAVE REQUESTED A PART NUMBER AND A SPECIFICATION PRINT FROM COMPONENTS ALREADY IN THE NORMAL MANNER. THE 120- NUMBER IN THIS SPACE MUST BE ONE FOR WHICH A VALID PART NUMBER AND STANDARD DESCRIPTION EXIST, AND FOR WHICH A SPECIFICATION DRAWING IS AT LEAST IN PROCESS IF NOT YET COMPLETED AND IN BLUEPRINT.

9. DECIDE WHAT YOU WANT THE STANDARD DESCRIPTION FOR THE 722- NUMBER TO BE. YOU MIGHT WANT TO HAVE IT BE THE SAME AS THE 122- NUMBER, BUT WITH A DIFFERENT ENDING (SUCH AS M FOR 'MACHINE CODE REPRESENTATION' OR X FOR 'ASCII-HEX REPRESENTATION'); OR YOU MAY INVENT A DIFFERENT NAME FOR IT. REMEMBER THAT YOU ARE GOING TO HAVE TO BE WORKING WITH THESE STANDARD DESCRIPTIONS, SO IT IS TO YOUR ADVANTAGE TO MAKE THEM EASILY DECODABLE. FILL IN THE STANDARD DESCRIPTION YOU HAVE CHOSEN FOR THE 722- NUMBER IN THE BOXES TO THE RIGHT OF THE BLANK SPACE BEGINNING WITH 722- .

10. FILL IN THE DATE YOU EXPECT THE 722- NUMBER TO BE RELEASED. NORMALLY, THE 722- RELEASE WILL OCCUR AS PART OF THE LOAD FILE RELEASE (SEE STAGE 2); THE LOAD FILE RELEASE OCCURS AT THE SAME TIME AS THE MANUFACTURING RELEASE OF YOUR PROJECT. THEREFORE, TO DETERMINE THE PROBABLE RELEASE DATE OF THE 722- DOCUMENT, TAKE THE DATE YOU EXPECT TO COMPLETE THE MANUFACTURING RELEASE AND ADD FIVE WORKING DAYS. THIS ALLOWS TIME FOR THE PROCESSING OF THE LOAD FILE RELEASE ORDER. HOWEVER, THIS MEANS THAT YOU WILL NOT BE ABLE TO WAIT UNTIL THE REST OF YOUR MANUFACTURING RELEASE IS COMPLETE BEFORE SUBMITTING THE LOAD FILE RELEASE ORDER; IT WILL NEED TO BE SUBMITTED AS PART OF THE MANUFACTURING RELEASE, THAT IS NEAR THE BEGINNING OF THE MANUFACTURING RELEASE PROCESS.

11. FILL IN THE DATE YOU EXPECT THE PURCHASING BILL OF MATERIALS TO BE RELEASED. SINCE THE BILL OF MATERIALS RELEASE AND THE PRELIMINARY SOFTWARE RELEASE ARE SUPPOSED TO BE CONCURRENT, THIS SHOULD BE A DATE WITHIN TWO WEEKS OF THE DATE YOU ARE COMPLETING THE PRELIMINARY SOFTWARE RELEASE. IF THE BILL OF MATERIALS IS ALREADY AVAILABLE FROM BLUEPRINT, FILL IN 'RELEASED'.

12. CHECK THE 'ELECTRICAL' OR 'MASK' SPACE DEPENDING ON THE TYPE OF PROGRAMMING OF THE DEVICE. FOR A NEW SYSTEM, THIS WILL NORMALLY BE 'ELECTRICAL'.

YOU HAVE NOW COMPLETED THE INFORMATION NECESSARY FOR A ONE-CHIP PROCESSOR. IF YOUR SYSTEM HAS MORE THAN ONE PROGRAMMABLE DEVICE, FILL IN ANOTHER SECTION AS ABOVE (NUMBERS 7 THROUGH 12) FOR EACH OF THE OTHER CHIPS INVOLVED.

YOU ARE NOW READY TO HAVE YOUR 122- AND 722- NUMBERS ASSIGNED. TAKE THE PRELIMINARY SOFTWARE RELEASE FORM TO THE ENGINEERING OFFICE AND ASK JAN ODELL TO ASSIGN A 722- NUMBER. SHE WILL WRITE THE NUMBER DOWN ON THE FORM AND MAKE A COPY OF THE FORM. SHE WILL THEN BE ABLE TO USE THE INFORMATION YOU HAD ALREADY FILLED IN TO PUT THE NEW NUMBER AND ITS STANDARD DESCRIPTION ON VAX.

NOW TAKE THE FORM TO COMPONENT ENGINEERING. ASK PJ TO ASSIGN A 122- NUMBER. SHE WILL WRITE THE NUMBER DOWN ON THE FORM AND MAKE A COPY OF THE FORM. SHE WILL THEN BE ABLE TO USE THE INFORMATION YOU HAD ALREADY FILLED IN TO MAKE SURE THAT THE 122- NUMBER AND ITS STANDARD DESCRIPTION ARE PUT ON VAX AND TO GENERATE A SPECIFICATION CONTROL DRAWING FOR THE 122- NUMBER. THE FACT THAT YOU REQUESTED THE 122- NUMBER CONSTITUTED A REQUEST FOR THE DRAWING; YOU DO NOT NEED TO FILL OUT AN ADDITIONAL FORM. THE COMPLETED SPECIFICATION DRAWING WILL BE PLACED IN BLUEPRINT WHERE IT WILL BE AVAILABLE TO PURCHASING DURING THEIR ASSESSMENT OF PIECE PART ORDERS CHANGED OR MADE NECESSARY BY THE PROJECT.

COMPONENT ENGINEERING WILL NOT ISSUE A 122- NUMBER UNLESS ALL THE OTHER INFORMATION ON THE PRELIMINARY RELEASE FORM (INCLUDING THE 722- NUMBER) IS ALREADY FILLED IN.

KEEP THE ORIGINAL OF THE PRELIMINARY RELEASE FORM IN YOUR RECORDS WITH YOUR SOFTWARE INITIATION FORM; YOU WILL NEED THE INFORMATION ON THEM LATER WHEN YOU DO THE LOAD FILE RELEASE.

STAGE 2: LOAD FILE RELEASE ORDER

WHEN THE TIME COMES FOR THE MANUFACTURING RELEASE OF YOUR PROJECT, REMEMBER THAT THE SOFTWARE RELEASE IS PART OF THE MANUFACTURING RELEASE. YOU BEGIN A SOFTWARE RELEASE BY FILLING OUT A LOAD FILE RELEASE ORDER AND SUBMITTING IT TO SOFTWARE CONTROL (KOREY WILLNAUER). SOFTWARE CONTROL WILL THEN USE THE INFORMATION ON THE LOAD FILE RELEASE ORDER TO CREATE MASTER CHIPS WHICH YOU WILL VERIFY BOTH ELECTRICALLY AND FUNCTIONALLY. RECEIVING INSPECTION WILL THEN USE THOSE MASTER CHIPS TO PROGRAM THE MICROPROCESSORS WHICH WILL BE USED IN PRODUCTION. IT IS THEREFORE A VERY GOOD IDEA FOR YOU TO FILL OUT THE LOAD FILE RELEASE ORDER COMPLETELY AND ACCURATELY. PLEASE FILL IT OUT AS FOLLOWS:

1. FILL IN THE 720- NUMBER OF YOUR PROCESSOR. THIS NUMBER WAS ISSUED TO YOU AT THE SAME TIME AS YOUR 719- AND 721- NUMBERS, AND IT SHOULD ALSO BE ON YOUR PRELIMINARY SOFTWARE RELEASE FORM.

2. FILL IN THE NAME OR NAMES OF THE SYSTEMS IN WHICH YOUR PROCESSOR APPEARS. A 'SYSTEM' IS A GROUP OF UNITS WHICH FUNCTION AS AN INTEGRAL WHOLE. FOR INSTANCE, THE KTR953, KAC952, AND KCU951 MAKE UP THE KMF 950 SYSTEM. SOME UNITS ARE THEIR OWN SYSTEM; FOR INSTANCE, THE KX 155/165 IS A SYSTEM UNTO ITSELF.

3. FILL IN THE NAME OF THE UNIT OR UNITS IN WHICH THE PROCESSOR IS USED. FOR INSTANCE, IF YOU FILLED IN 'KMF 950' UNDER 'SYSTEM', YOU MIGHT WRITE 'KAC952' HERE. IF THE 'SYSTEM' WAS 'KX 155/165', THE 'UNIT' WILL PROBABLY BE 'KX 155/165' TOO.

4. FILL IN THE PROCESSOR NAME. THIS WILL BE THE SAME NAME YOU DECIDED ON WHEN YOU FILLED IN THE PRELIMINARY SOFTWARE RELEASE.

5. FILL IN THE NAME OF YOUR SOURCE FILE OR FILES ON VAX. THESE WILL GENERALLY HAVE A NAME SOMETHING LIKE 'KY196V01.I48'.

6. FILL IN THE NAME WHICH YOU WANT TO BE USED FOR YOUR RELOCATABLE OBJECT FILES. THESE ARE THOSE INTERMEDIARY FILES WHICH OCCUR, FOR INSTANCE, IN A SITUATION WITH SEVERAL SOURCE FILES: AS EACH SOURCE FILE IS ASSEMBLED, AN 'OBJECT FILE' IS CREATED; ALL THE OBJECT FILES MUST THEN BE 'LINKED' TO CREATE A SINGLE DOWNLOADABLE 'LOAD FILE'. IF YOU HAVE ONLY ONE SOURCE FILE WHICH, WHEN ASSEMBLED, PRODUCES A SINGLE DOWNLOADABLE LOAD FILE, YOU MAY WRITE 'N/A' IN THIS OBJECT FILE SPACE.

7. FILL IN A NAME FOR THE COMMAND FILE WHICH SOFTWARE CONTROL WILL GENERATE FOR USE WITH YOUR SOURCE FILE. USUALLY THIS WILL BE THE SAME AS YOUR SOURCE FILE NAME BUT WITH A '.COM' EXTENSION. YOU MAY, HOWEVER, CHOOSE ANOTHER NAME IF YOU WISH.

8. FILL IN THE FORMAT OF THE PROGRAMMER WHICH YOU USED TO PROGRAM YOUR MICROPROCESSOR CHIPS. THERE ARE AT PRESENT PROGRAMMERS WITH THREE DIFFERENT FORMATS; THESE ARE

   A) INTEL
   B) MOTOROLA
   C) TEKTRONIX

FILL IN THE ONE OF THESE THREE TYPES WHICH YOUR PROGRAMMER USES.

9. FILL IN THE NAME WHICH YOU WANT TO BE USED FOR THE LOAD FILE WHICH SOFTWARE CONTROL WILL GENERATE. GENERALLY THIS WILL BE THE SAME AS YOUR SOURCE FILE NAME, BUT WITH A '.LOA' EXTENSION. YOU MAY, HOWEVER, CHOOSE ANOTHER NAME IF YOU WISH.

10. IF YOU ARE USING A 16-BIT DEVICE, FILL IN THE NAME OF YOUR EVEN-BYTE FILE. IF YOU ARE USING AN 8-BIT DEVICE, YOU MAY WRITE 'N/A' HERE.

11. IF YOU ARE USING A 16-BIT DEVICE, FILL IN THE NAME OF YOUR ODD-BYTE FILE. IF YOU ARE USING AN 8-BIT DEVICE, YOU MAY WRITE 'N/A' HERE.

12. FILL IN THE 122- NUMBER AND THE STANDARD DESCRIPTION WHICH YOU SHOULD HAVE ON YOUR COPY OF THE PRELIMINARY SOFTWARE RELEASE. ALTHOUGH THIS IS A LOAD FILE RELEASE FOR NEW SOFTWARE, THE 122- NUMBER AND ITS DESCRIPTION WERE ALREADY PUT ON VAX WHEN YOU DID THE PRELIMINARY SOFTWARE RELEASE; SO CHECK THE SPACE IN THE 'OLD' COLUMN NEXT TO THE 122- NUMBER.

13. FILL IN THE 120- NUMBER FOR THE UNPROGRAMMED DEVICE YOU ARE USING TO FOR YOUR PROGRAMMED CHIPS. IF THIS IS NOT THE SAME AS THE NUMBER ON THE PRELIMINARY SOFTWARE RELEASE, YOU'RE IN BIG TROUBLE. ALSO FILL IN THE TYPE NUMBER OF THE UNPROGRAMMED DEVICE. FOR INSTANCE, A 120-2129-01 IS A TYPE 8048.

14. FILL IN THE 722- NUMBER AND THE STANDARD DESCRIPTION WHICH YOU SHOULD HAVE ON YOUR COPY OF THE PRELIMINARY SOFTWARE RELEASE. ALTHOUGH THIS IS A LOAD FILE RELEASE FOR NEW SOFTWARE, THE 722- NUMBER AND ITS DESCRIPTION WERE ALREADY PUT ON VAX WHEN YOU DID THE PRELIMINARY SOFTWARE RELEASE; SO CHECK THE SPACE IN THE 'OLD' COLUMN NEXT TO THE 722- NUMBER.

15. FILL IN YOUR NAME, THE NAME OF THE PROJECT GROUP LEADER, AND THE NAME OF THE SOFTWARE CONTROL REPRESENTATIVE (KOREY WILLNAUER). LEAVE THE 'INITIALS' AND 'DATE' SECTIONS BLANK.

16. AT THE TOP OF THE SECOND PAGE, FILL IN THE 720- NUMBER AGAIN. (REMEMBER THAT THE 720- NUMBER IS THE NUMBER OF THE ACTUAL DOCUMENT YOU ARE FILLING IN, AS WELL AS OF THE PROCESSOR ITSELF.)

17. REFER BACK TO THE 'UNIT NAME (S)' SECTION ON PAGE ONE. YOU WILL NEED TO FILL IN AS MANY 715- NUMBERS AND STANDARD DESCRIPTIONS AS THERE ARE UNIT NAMES ON PAGE ONE. FILL IN THE 715- NUMBER MARKED 'A' WITH THE 715- NUMBER ASSIGNED TO THE UNIT IN THE UNIT NAME SPACE LABELED 'A' ON PAGE ONE. FILL IN THE ASSOCIATED STANDARD DESCRIPTION TO THE RIGHT OF THE 715- NUMBER. YOU SHOULD HAVE THIS INFORMATION ON THE 'UNIT LEVEL PART NUMBERS' FORM. ALTHOUGH THIS IS A LOAD FILE RELEASE FOR NEW SOFTWARE, THE 715- NUMBER AND ITS DESCRIPTION WERE ALREADY PUT ON VAX WHEN YOU DID THE PRELIMINARY SOFTWARE RELEASE; SO CHECK THE SPACE IN THE 'OLD' COLUMN NEXT TO THE 715- NUMBER.

FILL IN THE 715- NUMBERS LABELED 'B', 'C', AND 'D' ONLY IF YOU HAVE REFERRED TO THOSE UNITS IN THE 'UNIT NAME (S)' SECTION ON PAGE 1. BE SURE THAT THE 715- NUMBER YOU WRITE IN THE SPACE LABELED 'B' ON PAGE 2 CORRESPONDS TO THE UNIT IN THE SPACE LABELED 'B' ON PAGE 1, ETC.

CHECK THE SPACES IN THE 'OLD' COLUMN NEXT TO EACH FILLED-IN 715- NUMBER; (REMEMBER THAT ALTHOUGH THIS IS A RELEASE OF NEW SOFTWARE, THESE NUMBERS HAVE ALREADY BEEN ASSIGNED AND PUT ON VAX).

18. FILL IN THE 718- NUMBER ASSOCIATED WITH THIS PROCESSOR. THE 718- DOCUMENT IS A BIBLIOGRAPHY OF LITERATURE WHICH MIGHT BE USEFUL TO A PERSON WHO WAS TO BEGIN WORKING ON A PROJECT AFTER ITS INITIAL RELEASE. A STANDARD BIBLIOGRAPHY WITH AN ASSIGNED 718- NUMBER EXISTS FOR EACH TYPE OF PROGRAMMABLE DEVICE CURRENTLY STOCKED BY KING. THE ENGINEER SHOULD REFER TO THE APPROPRIATE DOCUMENT FOR THE DEVICE HE IS USING TO DETERMINE WHETHER HE WANTS TO USE THE STANDARD BIBLIOGRAPHY OR WHETHER HE SHOULD ASSIST SOFTWARE CONTROL IN PREPARING A SPECIAL BIBLIOGRAPHY FOR HIS MICROCOMPUTER. IF THE FORMER, HE SHOULD FILL IN THE STANDARD 718- NUMBER AND ITS STANDARD DESCRIPTION, AND CHECK THE SPACE IN THE 'OLD' COLUMN NEXT TO THE 718- NUMBER. IF THE LATTER, HE SHOULD LEAVE THE 718- NUMBER BLANK, CHECK THE 'NEW' COLUMN, AND FILL IN A SPECIFIC STANDARD DESCRIPTION. THE STANDARD DESCRIPTION MIGHT BE CONSTRUCTED AS FOLLOWS: 'UNIT NAME/ PROCESSOR NAME/ BIBLIO'. IT MUST NOT USE MORE THAN 18 CHARACTERS.

19. FILL IN THE 719-, 720-, AND 721- NUMBERS ASSOCIATED WITH THIS PROCESSOR, TOGETHER WITH THEIR STANDARD DESCRIPTIONS. THE 720- NUMBER IS THE SAME ONE YOU HAVE FILLED IN AS THE LOAD FILE RELEASE NUMBER. THE 719- AND 721- NUMBERS WERE ISSUED WHEN YOU SUBMITTED YOUR 'PROCESSOR LEVEL PART NUMBERS' FORM, AND SHOULD BE ON YOUR COPY OF THAT FORM. ALTHOUGH THIS IS A LOAD FILE RELEASE FOR NEW SOFTWARE, THESE NUMBERS AND THEIR DESCRIPTIONS WERE ALREADY PUT ON VAX WHEN YOU SUBMITTED THE PROCESSOR LEVEL FORM; SO CHECK THE SPACE IN THE 'OLD' COLUMN NEXT TO THE 122- NUMBER.

20. FILL IN THE 200- NUMBER OF THE BILL OF MATERIALS ON WHICH THE CHIP OR CHIPS WILL BE USED. IF IN A SINGLE-PROCESSOR, MULTIPLE-CHIP SYSTEM EACH CHIP IS USED ON A DIFFERENT BOARD AND IS THUS CALLED OUT ON A DIFFERENT BILL OF MATERIAL, REFERENCE ALL THE 200- NUMBERS HERE.

21. SINCE THIS IS A RELEASE OF NEW SOFTWARE, YOU WILL USUALLY BE ABLE TO WRITE N/A IN THE ECO AND EFF SPACES. HOWEVER, IF THERE IS AN ECO ASSOCIATED WITH YOUR SOFTWARE RELESE (IF, FOR INSTANCE, A NUMBER IS BEING PUT ON THE BILL OF MATERIALS FOR THE FIRST TIME OR IS BEING CHANGED FROM A 120- TO A 122- NUMBER) YOU MUST FILL IN THAT ECO NUMBER AND ITS EFFECTIVITY HERE.

22. FOR EACH PROGRAMMED DEVICE YOU DEFINED ON PAGE 1, YOU MUST FILL IN A SECTION HERE:

FILL IN THE DATE YOU EXPECT THE BINARY IMAGE FILE TO BE RELEASED. THE BINARY IMAGE FILE IS AN ASCII-HEX REPRESENTATION OF THE MACHINE CODE WHICH IS PROGRAMMED INTO THE CHIPS; SOFTWARE CONTROL GENERATES THIS DOCUMENT AFTER THE LOAD FILE RELEASE IS COMPLETED. THEREFORE, FILL IN A DATE TWO WEEKS FROM THE DATE YOU ARE GOING TO SUBMIT THE LOAD FILE RELEASE ORDER TO SOFTWARE CONTROL.

FILL IN THE ADDRESS IN THE LOAD FILE WHICH IS THE STARTING ADDRESS FOR THE CHIP. ALSO FILL IN THE CHIP SIZE IN HEX NOTATION. PLEASE DO NOT FILL IN '1K', ETC; USE '400 HEX' INSTEAD.

LEAVE THE .DOC AND .HEX FILE NAME SPACES BLANK.

23. IN THE 'SUPPORT SOFTWARE' SECTION, FILL IN THE NAMES AND VERSION NUMBERS OF THE SUPPORT SOFTWARE WHICH WILL BE NECESSARY FOR THE CONVERSION OF YOUR SOURCE FILE OR FILES INTO ONE OR MORE LOAD FILES SUITABLE FOR DOWNLOADING INTO PROGRAMMABLE DEVICES. FOR INSTANCE, THE NECESSARY COMMAND INTERPRETER ASSEMBLER MIGHT BE 01ASMBLR.EXE ; THE NECESSARY CONVERSION ASSEMBLER MIGHT BE 01ASM8048.EXE . YOU WILL ALWAYS SPECIFY A COMMAND INTERPRETER ASSEMBLER AND A CONVERSION ASSEMBLER; THE TYPE OF SOFTWARE WHICH YOU ARE GENERATING WILL DETERMINE WHETHER YOU NEED THE OTHER TYPES OF SUPPORT SOFTWARE. PLEASE WRITE 'N/A' IN THE SPACES NEXT TO THOSE TYPES OF SUPPORT SOFTWARE WHICH WILL NOT BE USED FOR YOUR PROJECT.

THE PARTITIONER AND FORMATTER ARE SOFTWARE PACKAGES RUN ON THE LOAD FILE TO OBTAIN A STANDARD FORMAT FOR DOWNLOADING INTO PROGRAMMABLE DEVICES. YOU MUST USE THE PARTITIONER AND FORMATTER WHEN YOU CREATE THE PROTOTYPE CHIPS TO BE USED IN TEST ENGINEERING UNITS; SPECIFY THE VERSIONS OF PARTITIONER AND FORMATTER WHICH YOU USED.

IF THIS STEP SEEMS PARTICULARLY COMPLICATED, PLEASE REFER TO APPENDIX A "XXX" FOR AN EXPLANATION OF THE ROLE OF THE DIFFERENT VARIETIES OF SUPPORT SOFTWARE AND OF YOUR RESPONSIBILITIES AS TO THEIR USE.

YOU HAVE NOW FILLED IN THE LOAD FILE RELEASE ORDER FOR YOUR PROCESSOR. PLEASE READ IT OVER TO MAKE SURE THAT YOU DID NOT LEAVE OUT ANY NECESSARY ITEM, AND THAT THE INFORMATION YOU HAVE LISTED IS ACCURATE AND ADEQUATE.

IF YOU ARE WRITING AN ECO WHICH IS RELATED TO THIS SOFTWARE RELEASE, BE SURE TO INCLUDE A NOTE ON THE ECO WHICH SPECIFIES WHAT IS TO BE DONE WITH ANY OLD PROGRAMMED PARTS.

NOW YOU ARE READY TO TRANSFER YOUR SOURCE FILE TO SILVER EXCHANGE; SOFTWARE CONTROL WILL COPY IT FROM THE EXCHANGE FILE AND USE IT TO CREATE DOWNLOADABLE FILES WHICH WILL THEMSELVES BE USED TO CREATE MASTER CHIPS. WHEN YOU TRANSFER YOUR SOURCE FILE, YOU NEED TO BE SURE THAT YOU GIVE SOFTWARE CONTROL THE PRIVILEGE TO DELETE IT WHEN THE CHIPS HAVE BEEN RELEASED AND THE RELEASE HAS BEEN ARCHIVED. THEREFORE, PLEASE PROCEED AS FOLLOWS:

IF YOU ARE ON NODE GOLD:

```
COPY/PROT=(W:RWED)
FROM:     [YOURDIRECTORYNAME]SOURCEFILENAME.EXT
TO:       SILVER::ENG$USERDISK:[EXCHANGE]SOURCEFILENAME.EXT
```

IF YOU ARE NODE SILVER:

PROCEED AS ABOVE BUT IN THE THIRD LINE BEGIN WITH [EXCHANGE].

NOW BRING THE LOAD FILE RELEASE ORDER TO SOFTWARE CONTROL. GIVE IT TO THE FRIENDLY SOFTWARE CONTROL REPRESENTTIVE. IF YOU HAVE ANY WORRIES OR QUESTIONS, NOW IS THE TIME TO VOICE THEM.

SOFTWARE CONTROL WILL REVIEW THE DOCUMENT AND CONSULT YOU IF THERE ARE ANY QUESTIONS, ESPECIALLY AS TO HOW TO DESIGN THE .COM FILE (WHICH WILL BE USED TO CREATE OBJECT, LOAD, .DOC, AND .HEX FILES). THE .DOC FILE IS A FORMATTED VERSION OF THE LOAD FILE WHICH IS OUR LEGAL REPRESENTATION OF WHAT WE AS A COMPANY PUT INTO OUT CHIPS. THE .HEX FILE IS AN ASCII-HEX REPRESENTATION OF THE MACHINE CODE IN A STANDARDIZED FORMAT.

SOFTWARE CONTROL WILL THEN ASK YOU TO CHECK THAT THE DOWNLOADABLE FILE IT OBTAINED IS IDENTICAL TO THE LOAD FILE THAT YOU USED TO PROGRAM TEST CHIPS. YOU WILL NORMALLY DO THIS BY RUNNING A 'DIFFERENCES' COMMAND ON THE TWO FILES, AND IF NECESSARY, BY CHECKING THAT ANY DIFFRENCES ARE DIFFERENCES OF FORM AND NOT OF SUBSTANCE.

SOFTWARE CONTROL WILL THEN CREATE MASTER CHIPS AND WILL ASK YOU TO 'VERIFY AND VALIDATE' THEM. THIS MEANS THAT YOU ARE EXPECTED TO CHECK THAT THE MASTER CHIP MATCHES YOUR TEST CHIP BOTH IN ELECTRICAL CONTENT AND IN QUALITY OF FUNCTIONING IN A WORKING UNIT.

WHEN THE LOAD FILE RELEASE IS COMPLETE AND ALL THE FILES ASSOCIATED WITH IT HAVE BEEN ARCHIVED, SOFTWARE CONTROL WILL INITIAL AND DATE THE LOAD FILE RELEASE ORDER, AND WILL ASK YOU AND YOUR GROUP LEADER TO DO THE SAME. YOUR DOING SO CONSTITUTES AN ACCEPTANCE OF RESPONSIBILITY FOR THE CONTENTS AND FUNCTIONING OF DEVICES WHICH ARE PUT INTO PRODUCTION, IN MUCH THE SAME WAY THAT YOUR SIGNING A TSO WOULD CONSTITUTE AN ACCEPTANCE OF RESPONSIBILITY FOR THAT DOCUMENT.

SOFTWARE CONTROL WILL THEN GIVE YOU A COPY OF THE COMPLETED LOAD FILE RELEASE ORDER. PLEASE KEEP THIS COPY IN YOUR RECORDS. IT WILL BE VERY HELPFUL TO YOU WHEN THE TIME COMES TO UPDATE THE SOFTWARE.

8: REVISIONS TO EXISTING SOFTWARE

WHEN YOU ARE GOING TO REVISE EXISTING SOFTWARE, AND YOUR NEW SOFTWARE IS COMPLETE AND WORKING, YOU NEED TO DO TWO THINGS: YOU NEED TO WRITE AN ECO; AND YOU NEED TO FILL OUT A LOAD FILE RELEASE ORDER.

THE ECO

THE ECO WILL BE WRITTEN TO THE BILL (OR BILLS) OF MATERIAL WHICH CALLS OUT THE CHIP OR CHIPS WHOSE SOFTWARE IS TO BE CHANGED. REMEMBER THAT WHEN YOU REVISE SOFTWARE, YOU DON'T TACK A REVISION NUMBER ONTO THE EXISTING PART NUMBER; YOU CREATE A NEW PART NUMBER WHICH IS IDENTICAL TO THE OLD PART NUMBER EXCEPT THAT THE LAST TWO DIGITS ARE INCREMENTED BY ONE. FOR INSTANCE, IF THE OLD PART NUMBER IS 122-0033-01, THE NEW PART NUMBER WILL AUTOMATICALLY BE 122-0033-02. THE NEXT TIME YOU REVISE THE SOFTWARE IN THAT CHIP, ITS NUMBER WILL BECOME 122-0033-03.

GET A COPY OF THE APPROPRIATE BILL OF MATERIAL. USING A RED PEN OR MARKER, CROSS OUT THE OLD 122- NUMBER AND FILL IN THE NEW ONE. (REMEMBER, THE NEW NUMBER IS THE SAME AS THE OLD ONE EXCEPT FOR THE LAST TWO DIGITS.) IF YOU ARE MAKING OTHER HARDWARE CHANGES TO THE ASSEMBLY DEFINED BY THIS BILL, YOU MAY MAKE THOSE CHANGES ON THIS ECO TOO. MARK OUT THE OLD INFORMATION AND FILL IN THE NEW.

WRITE THE ECO. INCLUDE A NOTE (EITHER IN THE 'COMMENTS' SECTION OR AT THE BOTTOM OF THE FORM) THAT DEFINES WHAT IS TO BE DONE WITH CHIPS IN STOCK THAT CONTAIN THE OLD PROGRAM. IF THE OLD PART WAS ELECTRICALLY PROGRAMMED, IT CAN USUALLY BE ERASED AND RE-USED. IF THE OLD PART WAS MASKED, IT MUST USUALLY BE SCRAPPED ACCORDING TO THE EFFECTIVITY OF THE ECO. THEREFORE, IF THE OLD PART WAS ELECTRICALLY PROGRAMMED, AND YOU ARE CHANGING 122-0033-01 TO 122-0033-02, THE NOTE ON YOUR ECO MIGHT LOOK LIKE THIS:

"122-0033-01 PARTS ARE ELECTRICALLY PROGRAMMED. THEY SHOULD BE ERASED AND RETURNED TO STOCK AS 120-0001-01 PARTS."

NOTICE THAT YOU ALSO SPECIFIED THE UNPROGRAMMED PART NUMBER OF THE CHIP.

IF THE OLD PART WAS MASKED, THE NOTE ON YOUR ECO MIGHT LOOK LIKE THIS:

"122-0033-01 PARTS ARE MASKED. THEY CANNOT BE ERASED AND MUST THEREFORE BE SCRAPPED."

THE LOAD FILE RELEASE ORDER

YOU SHOULD ALREADY BE SOMEWHAT FAMILIAR WITH THIS DOCUMENT AFTER USING IT TO RELEASE THE FIRST VERSION OF YOUR SOFTWARE. THE COPY OF THAT FIRST LOAD FILE RELEASE ORDER WILL BE VERY USEFUL TO YOU IN FILLING OUT THE FORM FOR THE NEW VERSION OF SOFTWARE; SO GO FIND IT.

REMEMBER THAT EVERY 122- NUMBER HAS A BINARY IMAGE FILE (AN ASCII-HEX REPRESENTATION) ASSOCIATED WITH IT. EACH BINARY IMAGE FILE HAS A 722- NUMBER. IT STANDS TO REASON THAT WHEN YOU CHANGE THE SOFTWARE IN A PROGRAMMED DEVICE, THE 122- NUMBER CHANGES. IT IS ALSO OBVIOUS THAT IF THE 122- NUMBER CHANGES, THE 722- NUMBER MUST ALSO CHANGE. THE LAST TWO DIGITS OF A 122- PROGRAMMED CHIP AND THE 722- NUMBER OF ITS BINARY IMAGE FILE SHOULD ALWAYS BE THE SAME.

FURTHER, A 720- DOCUMENT (A LOAD FILE RELEASE ORDER) RECORDS ALL OF THE PART NUMBERS ASSOCIATED WITH A LOAD FILE RELEASE. THEREFORE WHEN ANY ONE OR MORE NUMBERS ON THE 720- DOCUMENT CHANGE, A NEW 720- DOCUMENT WITH A NEW NUMBER (IDENTICAL TO THE OLD NUMBER EXCEPT FOR THE LAST TWO DIGITS) MUST BE GENERATED

REMEMBER THAT SOFTWARE CONTROL WILL BE USING THE INFORMATION ON THE LOAD FILE RELEASE ORDER TO CREATE MASTER CHIPS WHICH YOU WILL VERIFY BOTH ELECTRICALLY AND FUNCTIONALLY. RECEIVING INSPECTION WILL THEN USE THOSE MASTER CHIPS TO PROGRAM THE MICROPROCESSORS WHICH WILL BE USED IN PRODUCTION. IT IS THEREFORE A VERY GOOD IDEA FOR YOU TO FILL OUT THE LOAD FILE RELEASE ORDER COMPLETELY AND ACCURATELY.

KEEPING ALL OF THIS IN MIND, PLEASE FILL IN THE LOAD FILE RELEASE OREDER AS FOLLOWS:

1. FILL IN THE 720- NUMBER OF YOUR PROCESSOR. IT WILL BE THE SAME AS THE NUMBER ON THE FIRST LOAD FILE RELEASE YOU DID FOR THIS PROCESSOR, EXCEPT THAT THE LAST TWO DIGITS WILL BE INCREMENTED BY ONE. FOR INSTANCE, IF THE NUMBER ON THE PREVIOUS LOAD FILE RELEASE IS 720-0135-06, YOU SHOULD FILL IN 720-0135-07 ON THE NEW FORM.

2. FILL IN THE NAME OR NAMES OF THE SYSTEMS IN WHICH YOUR PROCESSOR APPEARS. A 'SYSTEM' IS A GROUP OF UNITS WHICH FUNCTION AS AN INTEGRAL WHOLE. FOR INSTANCE, THE KTR953, KAC952, AND KCU951 MAKE UP THE KHF 950 SYSTEM. SOME UNITS ARE THEIR OWN SYSTEM; FOR INSTANCE, THE KX 155/165 IS A SYSTEM UNTO ITSELF.

3. FILL IN THE NAME OF THE UNIT OR UNITS IN WHICH THE PROCESSOR IS USED. FOR INSTANCE, IF YOU FILLED IN 'KHF 950' UNDER 'SYSTEM', YOU MIGHT WRITE 'KAC952' HERE. IF THE 'SYSTEM' WAS 'KX 155/165', THE 'UNIT' WILL PROBABLY BE 'KX 155/165' TOO.

4. FILL IN THE PROCESSOR NAME. UNLESS YOU HAVE INCORPORATED A MAJOR CHANGE OF FUNCTION AND PURPOSE IN YOUR NEW SOFTWARE, YOU WILL PROBABLY WANT TO RETAIN THE SAME PROCESSOR SYSTEM NAME AS YOU HAD BEFORE. YOU MIGHT CONSIDER ADDING VERSION LEVEL INFORMATION TO THE NAME, AS IN CHANGING 'KY196/197 CONTROL PROCESSOR' TO 'KY196/197 CONTROL PROCESSOR V01'.

5. FILL IN THE NAME OF YOUR SOURCE FILE OR FILES ON VAX FOR THE NEW SOFTWARE. IT CANNOT BE THE SAME NAME AS THE SOURCE FILE NAME ON A PREVIOUS SOFTWARE RELEASE.

6. FILL IN THE NAME WHICH YOU WANT TO BE USED FOR YOUR RELOCATABLE OBJECT FILES. THESE ARE THOSE INTERMEDIARY FILES WHICH OCCUR IN A SITUATION WITH SEVERAL SOURCE FILES: AS EACH SOURCE FILE IS ASSEMBLED, AN 'OBJECT FILE' IS CREATED; ALL THE OBJECT FILES MUST THEN BE 'LINKED' TO CREATE A SINGLE DOWNLOADABLE 'LOAD FILE'. IF YOU HAVE ONLY ONE SOURCE FILE WHICH, WHEN ASSEMBLED, PRODUCES A SINGLE DOWNLOADABLE LOAD FILE, YOU MAY WRITE 'N/A' IN THIS OBJECT FILE SPACE. PLEASE DO NOT LEAVE THE SPACE BLANK.

7. FILL IN THE NAME WHICH YOU WANT TO BE USED FOR THE COMMAND FILE WHICH SOFTWARE CONTROL WILL GENERATE FOR USE WITH YOUR SOURCE FILE. USUALLY THIS WILL BE THE SAME AS YOUR SOURCE FILE NAME BUT WITH A '.COM' EXTENSION. YOU MAY, HOWEVER, CHOOSE ANOTHER NAME IF YOU WISH. EVENTUALLY THE ENGINEER WILL BE EXPECTED TO GENERATE A COMMAND FILE AND TRANSFER IT TO SOFTWARE CONTROL WITH HIS SOURCE FILE.

8. FILL IN THE FORMAT OF THE PROGRAMMER WHICH YOU USED TO PROGRAM YOUR MICROPROCESSOR CHIPS. THERE ARE AT PRESENT PROGRAMMERS WITH THREE DIFFERENT FORMATS; THESE ARE

   A) INTEL
   B) MOTOROLA
   C) TEKTRONIX

FILL IN THE ONE OF THESE THREE TYPES WHICH YOU USED.

9. FILL IN THE NAME WHICH YOU WANT TO BE USED FOR THE LOAD FILE WHICH SOFTWARE CONTROL WILL GENERATE. GENERALLY THIS WILL BE THE SAME AS YOUR SOURCE FILE NAME, BUT WITH A '.LOA' EXTENSION. YOU MAY, HOWEVER, CHOOSE ANOTHER NAME IF YOU WISH.

10. IF YOU ARE USING A 16-BIT DEVICE, FILL IN THE NAME OF YOUR EVEN-BYTE FILE. IF YOU ARE USING AN 8-BIT DEVICE, YOU MAY WRITE 'N/A' HERE.

11. IF YOU ARE USING A 16-BIT DEVICE, FILL IN THE NAME OF YOUR ODD-BYTE FILE. IF YOU ARE USING AN 8-BIT DEVICE, YOU MAY WRITE 'N/A' HERE.

12. LOOK AT THE SECTION LABELED '1' UNDER THE 'EVEN-BYTE FILE NAME' ON PAGE ONE OF THE PREVIOUS LOAD FILE RELEASE ORDER. IS THE SOFTWARE IN THIS DEVICE CHANGING?

IF IT IS, FILL IN THE NEW 122- AND 722- NUMBERS (THEY WILL BE THE SAME AS THE OLD NUMBERS, BUT INCREMENTED BY ONE). ALSO FILL IN THEIR NEW STANDARD DESCRIPTIONS. SINCE THESE ARE NEW NUMBERS NOT YET PUT ON VAX, CHECK THE SPACE IN THE COLUMN MARKED 'NEW' NEXT TO BOTH NUMBERS. ALSO FILL IN THE 120- NUMBER FOR THE UNPROGRAMMED DEVICE YOU ARE USING FOR YOUR PROGRAMMED CHIPS, TOGETHER WITH THE TYPE NUMBER OF THE UNPROGRAMMED DEVICE. FOR INSTANCE, A 120-2050-01 IS A TYPE 8048.

IF THE SOFTWARE IS NOT CHANGING, COPY THE NUMBERS AND STANDARD DESCRIPTIONS FROM THE OLD LOAD FILE RELEASE TO THE NEW ONE. SINCE THE NUMBERS HAVE ALREADY BEEN ISSUED AND PUT ON VAX, CHECK THE SPACE IN THE COLUMN MARKED 'OLD' NEXT TO BOTH NUMBERS.

13. REPEAT STEP 12 FOR EACH ADDITIONAL PROGRAMMABLE DEVICE IN YOUR MICROCOMPUTER. PLEASE MAKE SURE THAT YOU PUT THE INFORMATION IN SECTION '2' FOR THE SAME CHIP YOU HAD IN SECTION 2 ON THE ORIGINAL LOAD FILE RELEASE, ETC.

15. FILL IN YOUR NAME, THE NAME OF THE PROJECT GROUP LEADER, AND THE NAME OF THE SOFTWARE CONTROL REPRESENTATIVE (KOREY WILLNAUER). LEAVE THE 'INITIALS' AND 'DATE' SECTIONS BLANK.

16. AT THE TOP OF THE SECOND PAGE, FILL IN THE 720- NUMBER AGAIN. IT WILL BE THE SAME NUMBER THAT YOU FILLED IN ON PAGE 1, NOT THE NUMBER ON THE OLD LOAD FILE RELEASE. (REMEMBER THAT THE 720- NUMBER IS THE NUMBER OF THE ACTUAL DOCUMENT YOU ARE FILLING IN, AS WELL AS OF THE PROCESSOR ITSELF.)

17. REFER BACK TO THE 'UNIT NAME (S)' SECTION ON PAGE ONE. YOU WILL NEED TO FILL IN AS MANY 715- NUMBERS AND STANDARD DESCRIPTIONS AS THERE ARE UNIT NAMES ON PAGE ONE. DOES YOUR SOFTWARE CHANGE IMPLY A CHANGE IN THE UNIT LEVEL SOFTWARE DESCRIPTION DOCUMENT?

IF NOT, THE 715- NUMBER (S) WILL NOT CHANGE. COPY THE INFORMATION FROM THE OLD LOAD FILE RELEASE DOCUMENT, AND CHECK THE SPACE IN THE COLUMNMARKED 'OLD' NEXT TO THE 715- NUMBER.

HOWEVER, IF THE SOFTWARE CHANGE IS SO MAJOR THAT IT DOES IMPLY DIFFERENCES IN THE UNIT LEVEL SOFTWARE DESCRIPTION, THEN THE 715- NUMBER WILL BE INCREMENTED BY ONE. IN THIS CASE, WRITE DOWN THE NEW 715- NUMBER AND STANDARD DESCRIPTION, AND CHECK THE SPACE IN THE COLUMN MARKED 'NEW'.

MAKE SURE THAT YOU FILL IN THE 715- NUMBER MARKED 'A' WITH THE 715- NUMBER ASSIGNED TO THE UNIT YOU WROTE IN THE UNIT NAME SPACE LABELED 'A' ON PAGE ONE.

FILL IN THE 715- NUMBERS LABELED 'B', 'C', AND 'D' ONLY IF YOU HAVE REFERRED TO THOSE UNITS IN THE 'UNIT NAME (S)' SECTION ON PAGE 1. BE SURE THAT THE 715- NUMBER YOU WRITE IN THE SPACE LABELED 'B' ON PAGE 2 CORRESPONDS TO THE UNIT IN THE SPACE LABELED 'B' ON PAGE 1, ETC. FOLLOW THE SAME GUIDELINES AS ABOVE TO DECIDE WHETHER THE 715- NUMBERS SHOULD BE INCREMENTED BY ONE OR REMAIN THE SAME.

18. FILL IN THE 718- NUMBER ASSOCIATED WITH THIS PROCESSOR. THE 718- DOCUMENT IS A BIBLIOGRAPHY OF LITERATURE WHICH MIGHT BE USEFUL TO A PERSON WHO WAS TO BEGIN WORKING ON A PROJECT AFTER ITS INITIAL RELEASE. A STANDARD BIBLIOGRAPHY WITH AN ASSIGNED 718- NUMBER EXISTS FOR EACH TYPE OF PROGRAMMABLE DEVICE CURRENTLY STOCKED BY KING. THE ENGINEER SHOULD REFER TO THE APPROPRIATE DOCUMENT FOR THE DEVICE HE IS USING TO DETERMINE WHETHER HE WANTS TO USE THE STANDARD BIBLIOGRAPHY OR WHETHER HE SHOULD ASSIST SOFTWARE CONTROL IN PREPARING A SPECIAL BIBLIOGRAPHY FOR HIS MICROCOMPUTER. IF THE FORMER, HE SHOULD FILL IN THE STANDARD 718- NUMBER AND ITS STANDARD DESCRIPTION, AND CHECK THE SPACE IN THE 'OLD' COLUMN NEXT TO THE 718- NUMBER. IF THE LATTER, HE SHOULD LEAVE THE 718- NUMBER BLANK, CHECK THE 'NEW' COLUMN, AND FILL IN A SPECIFIC STANDARD DESCRIPTION. THE STANDARD DESCRIPTION MIGHT BE CONSTRUCTED AS FOLLOWS: 'UNIT NAME/ PROCESSOR NAME/ BIBLIO'. IT MUST NOT USE MORE THAN 18 CHARACTERS.

19. FILL IN THE 719- NUMBER AND STANDARD DESCRIPTION ASSOCIATED WITH THIS RELEASE. SINCE THE 719- DOCUMENT IS THE SOURCE LISTING DOCUMENT, AND SINCE IF YOU ARE REVISING THE SOFTWARE YOU MUST HAVE CHANGED THE SOURCE CODE, THIS NUMBER WILL ALWAYS BE INCREMENTED BY ONE FROM WHAT IT WAS ON THE LAST LOAD FILE RELEASE. THE NEW STANDARD DESCRIPTION, TOO, SHOULD REFLECT THE REVISION LEVEL OF THE SOFTWARE. SINCE THIS IS A NEW NUMBER NOT YET DEFINED ON VAX, CHECK THE SPACE IN THE COLUMN MARKED 'NEW' NEXT TO THE 719- NUMBER.

20. FILL IN THE 720- NUMBER ASSOCIATED WITH THIS PROCESSOR, TOGETHER WITH ITS STANDARD DESCRIPTION. THE 720- NUMBER IS THE SAME ONE YOU HAVE FILLED IN AS THE LOAD FILE RELEASE NUMBER; ITS STANDARD DESCRIPTION WILL PROBABLY BE THE SAME AS THE ONE ON THE OLD LOAD FILE RELEASE EXCEPT THAT THE REVISION LEVEL WILL INCREASED BY ONE. FOR INSTANCE, IF THE OLD STANDARD DESCRIPTION WAS 'RADIAL SYNTHESIZER V02', THE NEW STANDARD DESCRIPTION MIGHT BE 'RADIAL SYNTHESIZER V03'. HOWEVER, IF THE FUNCTION OF THE MICROCOMPUTER HAS CHANGED SIGNIFICANTLY, YOU SHOULD CONSIDER CHANGING ITS NAME ALTOGETHER. SINCE THIS IS A NEW NUMBER NOT YET DEFINED ON VAX, CHECK THE SPACE IN THE COLUMN MARKED 'NEW' NEXT TO THE 719- NUMBER.

21. FILL IN THE 721- NUMBER AND STANDARD DESCRIPTION ASSOCIATED WITH THIS LOAD FILE RELEASE. SINCE YOUR SOURCE FILE HAS CHANGED, THE DESIGN DESCRIPTION DOCUMENT (AND THEREFORE THE 721- NUMBER) MUST ALSO NECESSARILY CHANGE. DEPENDING ON THE EXTENT OF THE SOFTWARE REVISION, YOU MAY CHOOSE TO KEEP THE SAME STANDARD DESCRIPTION AS BEFORE WITH AN AUGMENTED REVISION LEVEL; OR YOU MAY PREFER TO CHOOSE A DIFFERENT STANDARD DESCRIPTION ENTIRELY. SINCE THIS IS A NEW NUMBER NOT YET DEFINED ON VAX, CHECK THE SPACE IN THE COLUMN MARKED 'NEW' NEXT TO THE 721- NUMBER.

22. FILL IN THE 200- NUMBER OF THE BILL OF MATERIALS ON WHICH THE CHIP OR CHIPS WILL BE USED. IF IN A SINGLE-PROCESSOR, MULTIPLE-CHIP SYSTEM EACH CHIP IS USED ON A DIFFERENT BOARD AND IS THUS CALLED OUT ON A DIFFERENT BILL OF MATERIAL, REFERENCE ALL THE 200- NUMBERS HERE.

23. YOU MAY LEAVE THE ECO # AND EFFECTIVITY SECTIONS BLANK; THEY WILL BE FILLED IN BY THE ECO REPRESENTATIVE WHEN YOU SUBMIT YOUR ECO AND THE LOAD FILE RELEASE TO HIM.

24. FOR EACH PROGRAMMED DEVICE YOU DEFINED ON PAGE 1, YOU MUST FILL IN A SECTION HERE. FILL IN THE ADDRESS IN THE LOAD FILE WHICH IS THE STARTING ADDRESS FOR THE CHIP. ALSO FILL IN THE CHIP SIZE IN HEX NOTATION. PLEASE DO NOT FILL IN '1K', ETC; USE '400 HEX' INSTEAD. LEAVE THE .DOC AND .HEX FILE NAME SPACES BLANK.

23. IN THE 'SUPPORT SOFTWARE' SECTION, FILL IN THE NAMES AND VERSION NUMBERS OF THE SUPPORT SOFTWARE WHICH WILL BE NECESSARY FOR THE CONVERSION OF YOUR SOURCE FILE OR FILES INTO ONE OR MORE LOAD FILES SUITABLE FOR DOWNLOADING INTO PROGRAMMABLE DEVICES. FOR INSTANCE, THE NECESSARY COMMAND INTERPRETER ASSEMBLER MIGHT BE 01ASM8LR.EXE ; THE NECESSARY CONVERSION ASSEMBLER MIGHT BE 01ASM8048.EXE . YOU WILL ALWAYS SPECIFY A COMMAND INTERPRETER ASSEMBLER AND A CONVERSION ASSEMBLER; THE TYPE OF SOFTWARE WHICH YOU ARE GENERATING WILL DETERMINE WHETHER YOU NEED THE OTHER TYPES OF SUPPORT SOFTWARE. PLEASE WRITE 'N/A' IN THE SPACES NEXT TO THOSE TYPES OF SUPPORT SOFTWARE WHICH WILL NOT BE USED FOR YOUR PROJECT.

THE PARTITIONER AND FORMATTER ARE SOFTWARE PACKAGES RUN ON THE LOAD FILE TO OBTAIN A STANDARD FORMAT FOR DOWNLOADING INTO PROGRAMMABLE DEVICES. YOU MUST USE THE PARTITIONER AND FORMATTER WHEN YOU CREATE THE PROTOTYPE CHIPS TO BE USED IN TEST ENGINEERING UNITS; SPECIFY THE VERSIONS OF PARTITIONER AND FORMATTER WHICH YOU USED.

IF THIS STEP SEEMS PARTICULARLY COMPLICATED, PLEASE REFER TO APPENDIX A "XXX" FOR AN EXPLANATION OF THE ROLE OF THE DIFFERENT VARIETIES OF SUPPORT SOFTWARE AND OF YOUR RESPONSIBILITIES AS TO THEIR USE.

YOU HAVE NOW FILLED IN THE LOAD FILE RELEASE ORDER FOR YOUR PROCESSOR WITH ITS NEW SOFTWARE.
PLEASE READ IT OVER TO MAKE SURE THAT YOU DID NOT LEAVE OUT ANY NECESSARY ITEM, AND THAT THE INFORMATION YOU HAVE LISTED IS ACCURATE AND ADEQUATE.
ATTACH THE LOAD FILE RELEASE ORDER TO THE ECO (REMEMBER TO FOLLOW THE INSTRUCTIONS OUTLINED ABOVE FOR THE ECO) TOGETHER WITH ANY OTHER MARKED PRINTS AND SUBMIT THEM TO THE ECO REPRESENTATIVE. THE ECO REPRESENTATIVE WILL NOT ACCEPT AN ECO WHICH CHANGES THE REVISION LEVEL OF A 122- NUMBER UNLESS A LOAD FILE RELEASE ORDER IS SUBMITTED WITH IT.
SOFTWARE METHODOLOGY STEVE RUSSELL
SILVER::ENG:[SOFTLIB.DOC.NEWTABLE] 15 JUNE 1983

8888888888888888888888888888888888888888888888888888888888888888888888888888888888
TEXT BELOW WRITTEN BY DEBBIE KEITER:

****
EXPLANATION OF BASIC SUPPORT SOFTWARE
****

SECTION A: SUPPORT SOFTWARE

**888
****
2: WHICH VERSION OF SUPPORT SOFTWARE ARE YOU USING?
****

WHEN CREATING YOUR COMMAND FILE AND WHEN FILLING IN THE LOAD FILE RELEASE ORDER, YOU WILL NEED TO BE AWARE OF WHAT VERSIONS OF SUPPORT SOFTWARE YOU ARE USING. YOU DON'T ACTUALLY NEED TO KNOW WHAT VERSION YOU ARE USING IN ORDER TO USE IT; YOU JUST NEED TO KNOW SO THAT YOU CAN WRITE IT DOWN. REMEMBER, ONE OF THE PRIME OBJECTIVES OF SOFTWARE CONTROL IS TO DOCUMENT THE CREATION OF SOFTWARE COMPLETELY AND THOROUGHLY. WRITING DOWN THE VERSION OF SUPPORT SOFTWARE USED TO CREATE A GIVEN VERSION OF TARGET (DOWNLOADABLE) SOFTWARE ENSURES THAT THAT SAME VERSION OF TARGET SOFTWARE CAN ALWAYS BE RECREATED EXACTLY NO MATTER HOW MUCH THE CONVENTIONS USED IN THE SUPPORT SOFTWARE CHANGE.

ALL THE CURRENT VERSIONS OF SUPPORT SOFTWARE ARE CONTAINED ON DISK DRA0 IN A DIRECTORY CALLED [MICROLIB]. IF, WHEN LOGGED ON TO A VAX TERMINAL, YOU TYPE $DIR DRA0:[MICROLIB], THE TERMINAL WILL DISPLAY A LIST OF FILE NAMES SOMETHING LIKE THIS:

00ASM6800.EXE;1	00ASM8051.EXE;1	00ASM8068.EXE;1
00LOA6800.EXE;1	00LOA8051.EXE;1	00LOADER.EXE;1
00LOAZ80.EXE;1	00SEPARAT.EXE;1	01ASM8048.EXE;1
01ASM8LR.EXE;1	02FORMAT.EXE;1	02PART.EXE;1
03FORMAT.EXE;1	MACROLIB.Z8K;33	MICROINST.COM;1
MICROSYM.COM;2	MPASCAL.EXE;1	PASCAL.COM;2

RUNLIB.MPO;5 SIM8048.EXE;1 SIM8051.EXE;1
SIMULATE.EXE

THIS, IN EFFECT, IS A LIST OF CURRENT SUPPORT SOFTWARE. IT TELLS YOU WHICH VERSIONS YOU ARE USING WHEN YOU USE THE COMMAND 'ASSE', FOR INSTANCE. YOU ONLY NEED TO ABLE TO RECOGNIZE WHICH FILENAME CORRESPONDS TO WHICH SOFTWARE, AND YOU CAN COPY THE FILENAME DIRECTLY FROM HERE TO THE LOAD FILE RELEASE ORDER.

AS A RULE, THE COMMAND INTERPRETERS HAVE NAMES CORRESPONDING CLOSELY TO THEIR FUNCTION, AND DO NOT CONTAIN ANY DEVICE-SPECIFIC INFORMATION: THE 'ASSEMBLER' COMMAND INTERPRETER IS 01ASM8LR.EXE. NOTE THAT THERE IS ONLY ONE FILE IN THE LIST ABOVE WITH THE ASM8LR DESIGNATION. SIMILARLY, 00LOADER.EXE (THE LINKER OR 'LINKING LOADER') IS THE 'LINKER; COMMAND INTERPRETER. THE COMPILER, SINCE IT IS SPECIFIC TO A GIVEN HIGH-LEVEL LANGUAGE, IS RECOGNIZED BY THE NAME OF THAT LANGUAGE AND THE .COM EXTENSION: PASCAL.COM IS THE 'COMPILER' COMMAND INTERPRETER FOR PASCAL.

THE 'CONVERSION' SUPPORT SOFTWARE FOR THE COMPILER WILL AGAIN INDICATE THE TYPE OF HIG-LEVEL LANGUAGE, BUT IT WILL USUALLY HAVE THE .EXE EXTENSION. MPASCAL.EXE IS THE 'CONVERSION' COMPILER FOR PASCAL. THE MACRO LIB, YOU WILL REMEMBER, IS SPECIFIC TO A CERTAIN DEVICE TYPE, SO YOU SHOULD LOOK FOR THAT INFORMATION IN THE FILEAME: THE MACRO LIB FOR THE Z8K (???) CHIP IS MACROLIB.Z8K. THERE IS ONLY ONE CURRENT RUN LIB (??) AT A TIME; ACCORDING TO THE ABOVE LIST, IT MUST BE RUNLIB.MPO (DO WE INCLUDE VERSION LEEL NUMBERS?).

FOR THE ASSEMBLER, REMEMBER THAT THE CONVERSION SUPPORT SOFTWARE WILL AGAIN BE DEVICE-SPECIFIC. THERE IS ONLY ONE COMMAND INTERPRETER LISTED, AND THAT IS 01ASM8LR.EXE; BUT IF YOU ARE PROGRAMMING AN 8748 CHIP, YOU WILL USE THE 01ASM8048.EXE CONVERSION ASSEMBLER, WHILE IF YOU ARE PROGRAMMING AN 8751 YOU WILL THE 00ASM8051.EXE CONVERSION ASSEMBLER. (THE SUPPORT SOFTWARE GENERALLY CALLS OUT THE NUMBER FOR THE BASE ELECTRICAL UNMASKED PART RATHER THAN FOR THE EPROM.)

THE LINKER IS SIMILARLY DEVICE-SPECIFIC. REMEMBER, HOWEVER, THAT IN MACROLIB THE LINKERS ARE USUALLY REFERRED TO AS 'LOADERS'. THEREFORE, THE CONVERSION LINKER FOR THE 6809 IS 00LOA6809.EXE; FOR THE 8751, IT IS 00LOA8051.EXE. THESE ARE THE NAMES THAT YOU WILL FILL IN THE 'LINKER' SPACE OF THE 'CONVERSION' SECTION OF THE LOAD FILE RELEASE ORDER (ASSUMING THAT YOUR SOFTWARE REQUIRES THE USE OF A LINKER).

THE OUTPUT SUPPORT SOFTWARE PACKAGES HAVE, AS A RULE, ONLY ONE VERSION EACH. THUS THE CURRENT SEPARATOR IN THE EXAMPLE ABOVE IS 00SEPARAT.EXE; THE CURRENT PARTITIONER IS 02PART.EXE; AND THE CURRENT FORMATTER IS 03FORMAT.EXE. NOTE THAT THERE ARE TWO DIFFERENT VERSIONS LISTED OF THE FORMATTER: THE 02FORMAT.EXE AND THE 03FORMAT.EXE. IN CASES SUCH AS THIS, USE THE HIGHEST VERSION NUMBER ON THE LOAD FILE RELEASE ORDER (IN THIS CASE, THE 03 RATHER THAN THE 02 FORMATTER).

THE 'ARCHIVES' SECTION IS FOR SOFTWARE CONTROL'S RECORDS; THE SOFTWARE PROJECT ENGINEER NEED NOT FILL IT IN.

REMEMBER THAT THE VERSIONS OF SUPPORT SOFTWARE WHICH YOU WRITE DOWN ON THE LOAD FILE RELEASE ORDER MUST BE THE SAME VERSIONS YOU USED WHEN YOU RAN YOUR COMMAND FILE TO CREATE YOUR OWN .LOA, .DOC, AND .HEX FILES. THIS MEANS THAT YOU MUST LOOK UP THE CURRENT VERSIONS OF SUPPORT SOFTWARE WHEN YOU RUN THE COMMAND FILE, AND NOT WHEN YOU ARE READY TO SUBMIT THE SOFTWARE RELEASE TO SOFTWARE CONTROL.

NOW YOU ARE READY TO TRANSFER YOUR SOURCE AND COMMAND FILES TO SILVER EXCHANGE; SOFTWARE CONTROL WILL COPY THEM FROM THE EXCHANGE FILE AND USE IT TO CREATE DOWNLOADABLE FILES WHICH WILL THEMSELVES BE USED TO CREATE MASTER CHIPS. SEE APPENDIX 2 IF YOU ARE UNCERTAIN AS TO THE REQUIREMENTS FOR THE COMMAND FILE.

WHEN YOU TRANSFER YOUR SOURCE FILE, YOU NEED TO BE SURE THAT YOU GIVE SOFTWARE CONTROL THE PROVILEGE TO DELETE IT WHEN THE CHIPS HAVE BEEN RELEASED AND THE RELEASE HAS BEEN ARCHIVED. THEREFORE, PLEASE PROCEED AS FOLLOWS:

IF YOU ARE ON NODE GOLD:

```
COPY/PROT=(W:RWED)
FROM:     [YOURDIRECTORYNAME]SOURCEFILENAME.EXT
TO:       SILVER::ENG$USERDISK:[EXCHANGE]SOURCEFILENAME.EXT
```

IF YOU ARE NODE SILVER:

PROCEED AS ABOVE BUT IN THE THIRD LINE BEGIN WITH (?)

NOW BRING THE LOAD FILE RELEASE ORDER TO SOFTWAR CONTROL. GIVE IT TO THE FRIENDLY SOFWARE CONTROL REPRESENTTIVE. IF YOU HAVE ANY WORRIES OR QUESTIONS, NOW IS THE TIME TO VOICE THEM.
SOFTWARE
CONTROL WILL REVIEW THE DOCUMENT AND CONSULT YOU IF THERE ARE ANY QUESTIONS, ESPECIALLY AS TO HOW TO DESIGN THE .COM FILE (WHICH WILL BE USED TO CREATE OBJECT, LOAD, .DOC, AND .HEX FILES).
SOFTWARE CONTROL WILL THEN
ASK YOU TO CHECK THAT THE DOWNLOADABLE FILE IT OBTAINED IS IDENTICAL TO THE LOAD FILE THAT YOU USED TO PROGRAM TEST CHIPS. YOU WILL NORMALLY DO THIS BY RUNNING A 'DIFFERENCES' COMMAND ON THE TWO FILES, AND IF NECESSARY, BY CHECKING THAT ANY DIFFRENCES ARE DIFFERENCES OF FORM AND NOT OF SUBSTANCE.

SOFTWARE CONTROL WILL THEN CREATE MASTER CHIPS AND WILL ASK YOU TO 'VERIFY AND VALIDATE' THEM. THIS MEANS THAT YOU ARE EXPECTED TO CHECK THAT THE MASTER CHIP MATCHES YOUR TEST CHIP BOTH IN ELECTRICAL CONTENT AND IN QUALITY OF FUNCTIONING IN A WORKING UNIT.

WHEN THE LOAD FILE RELEASE IS COMPLETE AND ALL THE FLES ASSOCIATED WITH IT HAVE BEEN ARCHIVED, SOFTWARE CONTROL WILL
INITIAL AND DATE THE LOAD FILE RELEASE ORDER, AND WILL ASK YOU AND YOUR GROUP LEADER TO DO THE SAME. YOUR DOING SO CONSTITUTES
AN ACCEPTANCE OF RESPONSIBILITY FOR THE CONTENTS AND FUNCTIONING OF DEVICES WHICH ARE PUT INTO PRODUCTION, IN MUCH THE SAME WAY THAT YOUR SIGNING A TSO WOULD CONSTITUTE AN ACCEPTANCE OF RESPONSIBILITY FOR THAT DOCUMENT.

SOFTWARE CONTROL WILL THEN GIVE YOU A COPY OF THE COMPLETED LOAD FILE RELEASE ORDER. PLEASE KEEP THIS COPY IN YOUR RECORDS. IT WILL BE VERY HELPFUL TO YOU WHEN THE TIME COMES TO UPDATE THE SOFTWARE.

MEMO TO: JERRY WROBLEWSKI
FROM: STEVE RUSSELL

DATE: 10 JUNE 1982

SUBJECT: GETTING SOFTWARE INTO PRODUCTION.

I NOW HAVE A PROCEDURE FOR STARTUPS THAT CAN HELP IN THE TRANSITION FROM THE OLD SOFTWARE RELEASE PROCEDURE TO THE NEW ONE. FOR NOW, THE DOCUMENTATION WILL HAVE TO LAG THE ACTUAL RELEASE BUT THIS WILL NOT BE THE CASE FOR NEWER PROJECTS.

THERE ARE CURRENTLY THREE STAGES OF SOFTWARE RELEASE:

1. TOP-LEVEL SOFTWARE.
2. LRU SOFTWARE DOCUMENTATION.
3. LOAD FILE SOFTWARE.

YOUR PROJECT HAS PROGRESSED WELL BEYOND THE TIME FOR THE FIRST TWO RELEASES AND YOU ARE READY FOR THE THIRD RELEASE WHICH SHOULD COME A MONTH OR TWO BEFORE THE PROGRAMMED PARTS ARE ACTUALLY NEEDED.

FOR THE PRESENT TIME, YOU FIRST NEED TO IDENTIFY EACH PROGRAMMED I.C. THAT WILL BE USED IN A PARTICULAR UNIT AND OBTAIN A 122-XXXX-RN PART NUMBER ASSIGNMENT FOR THE PROGRAMMED ELECTRICAL PART. AN APPLICATION FORM IS ENCLOSED AND ALL COPIES SHOULD BE SUBMITTED TO COMPONENT ENGINEERING. NEXT YOU NEED TO GET A 722-XXXX-RN DOCUMENT NUMBER ASSIGNED TO THE BINARY IMAGE FILE THAT WILL BE PROGRAMMED INTO EACH CHIP. A COPY OF THE FORM NEEDED FOR THIS IS ALSO ENCLOSED. AFTER YOU HAVE A VALID 722-XXXX-RN NUMBER ENTERED IN THE SYSTEM, YOU CAN REQUEST COMPONENT ENGINEERING TO RELEASE THE 122-XXXX-RN DOCUMENT INTO THE SYSTEM. THEY CANNOT DO A RELEASE WITHOUT THE 722-XXXX-RN NUMBER.

WHEN THE ACTUAL TARGET SOFTWARE IS READY TO BE RELEASED INTO REVISION CONTROL, YOU WILL NEED TO DO A STAGE-3 RELEASE OF THE LOAD FILE SOFTWARE. THIS RELEASE REQUIRES THE INFORMATION AND PART NUMBER ASSIGNMENTS CALLED OUT IN THE ENCLOSED FORM.

AFTER THE INFORMATION ON THE FORM IS COMPLETED, I WILL ACT AS SOFTWARE CONTROL AND WORK WITH THE SOFTWARE PROJECT ENGINEER TO CREATE THE LOAD FILE AND BINARY IMAGE FILES ON VAX. THESE WILL BE ARCHIVED AND THE SOURCE FILES CAN ONLY BE UPDATED BY ECO.

THE NEXT STEP WILL BE THE PROGRAMMING OF THE MASTER COPIES OF ALL 122-XXXX-RN DEVICES WITH THEIR CORRESPONDING SOFTWARE. I WILL WORK WITH THE PE TO MAKE FOUR (4) MASTER COPIES OF EACH UV PROM. THE PROJECT ENGINEER WILL TAKE ONE OF THE COPIES AND USE IT FOR VERIFICATION AGAINST THE SAME CHIP THAT HE USES IN THE TESTED ENGINEERING MODEL. THE PE WILL BE REQUIRED TO SIGN OFF ON THE RELEASE FORM THAT HE HAS COMPLETED THIS VERIFICATION PROCESS.

AFTER THE VERIFICATION PROCESS HAS SHOWN THAT THE CHIPS CONTAIN THE DESIRED SOFTWARE, THE THREE REMAINING MASTER COPIES WILL BE TURNED OVER TO DALE COOPER SO THAT HE WILL HAVE THEM ON HAND WHEN RECEIVING INSPECTION NEEDS THEM.

WHEN REVISIONS ARE MADE TO THE SOURCE CODE, THE LOAD FILE RELEASE PROCEDURE IS REPEATED AND THE DASH NUMBER ON AFFECTED SOFTWARE DOCUMENTS AND PART NUMBERS IS INCREMENTED.

AFTER DALE COOPER HAS THE MASTER COPIES AVAILABLE, THE PRODUCTION PARTS (FOR UV PROMS) ARE PROGRAMMED AS FOLLOWS:

1. RECEIVING INSPECTION (DENNY HALE) GETS A WORK ORDER FROM PURCHASING PLANNER. THE WORK ORDER REQUESTS A QUANTITY OF PARTS TO BE WORKED UP PER A 122-XXXX-RN DRAWING. THE PLANNER ALSO PROVIDES R.I. WITH THE UNPROGRAMMED 120-XXXX-XX PARTS.

2. R.I. GETS A MASTER CHIP FROM DALE COOPER (COMPONENT ENGINEERING) TO BE USED AS THE SOFTWARE SOURCE. THE CHIPS ARE PROGRAMMED ON THE GANG PROGRAMMER THAT FITS THE DATA I/O PROGRAMMER IN HANNEMAN'S LAB. THEY ARE THEN STOCKED UNDER THE 122-XXXX-RN PART NUMBER THAT APPEARS ON THE B/M.

NOTICE THAT ONCE THE PROJECT ENGINEER HAS HELPED SOFTWARE CONTROL TO PRODUCE THE THREE MASTER COPIES, ENGINEERING IS NO LONGER INVOLVED IN THE PROGRAMMING OF CHIPS. COMPONENT ENGINEERING ALSO HAS EXTRA COPIES OF THE MASTER CHIP AVAILABLE SO THAT PROGRAMMED PARTS CAN BE INDEPENDENTLY VERIFIED BY A CHIP OTHER THAN THE ONE THAT THEY WERE PROGRAMMED WITH.

IN THE FUTURE, WE HOPE TO HAVE RECEIVING INSPECTION PEOPLE TRAINED TO DOWNLOAD THE SPECIFIED 722-XXXX-RN FILES DIRECTLY FROM VAX AND PROGRAM

THE NECESSARY PARTS.

COPIES: GARY BURRELL
DALE COOPER
DAN HARDER
DENNY HALE
TOM DENNIS

SOFTWARE METHODOLOGY — STEVE RUSSELL
DRA0:[STEVE.SOFTJMEM02.LIS — 18 JUNE 1982
REV:

MEMO TO: SOFTWARE ENGINEERS — DATE: 18 JUNE 1982
FROM: STEVE RUSSELL X2505

SUBJECT: FIRST DRAFT OF SOFTWARE RELEASE AND DOCUMENTATION PROCEDURES.

NEW REQUIREMENTS FOR SOFTWARE CONFIGURATION MANAGEMENT, SUCH AS DO-178, AND RECENT INCREASES IN THE SIZE AND COMPLEXITY OF SOFTWARE PRODUCED AT KING RADIO ARE CREATING THE NEED TO HAVE MORE FORMAL SOFTWARE RELEASE AND DOCUMENTATION METHODS. AT THE PRESENT TIME, JOHN CAROCARI, TOM DENNIS, AND I ARE INVOLVED IN A MAJOR EFFORT TO DEVELOP A COMPLETE SOFTWARE METHODOLOGY FOR KING RADIO. THIS EFFORT ENCOMPASSES ALL PHASES OF SOFTWARE SPECIFICATION, DESIGN, DEVELOPMENT, TESTING, DOCUMENTATION, PRODUCTION, AND MAINTENANCE.

THIS MEMO WILL ADDRESS IMMEDIATE NEEDS TO HAVE A DEFINITION OF SOFTWARE RELEASE AND DOCUMENTATION PROCEDURES. THESE IDEAS ARE STILL IN THE FORMULATION STAGE SO THIS MEMO MUST BE CONSIDERED HIGHLY PRELIMINARY.

ENCLOSED ARE COPIES OF THE VARIOUS FORMS DEVELOPED THUS FAR. THESE WILL BE REFERED TO IN THE TEXT OF THIS MEMO.

THE BASIC IDEA BEHIND THE RELEASE STRATEGY THAT WILL BE DESCRIBED IS TO HAVE THE RELEASES PHASED INTO SEVERAL STAGES. THIS PROVIDES A WAY FOR PROJECT MANAGEMENT TO HAVE CONTINUING VISIBILITY ON PROJECT STATUS AND, AT THE SAME TIME, THE DOCUMENTATION EFFORT CAN BE SPREAD OUT OVER THE SAME PERIOD AS THE DESIGN AND DEVELOPMENT. THE FINAL RELEASE OF ACTUAL CODE IS SCHEDULED AS CLOSE TO THE END OF THE DESIGN AND DEVELOPMENT CYCLE AS POSSIBLE. THIS IS DESIRABLE BECAUSE IT AVOIDS THE DOCUMENTATION BURDEN OF REVISION CONTROL DURING THE HIGHLY FLUID DEVELOPMENT PHASE WITHOUT COMPROMISING DESIRABLE CONFIGURATION MANAGEMENT GOALS.

AT THE BEGINNING OF A PROJECT, THE PROJECT ENGINEERING MANAGER AND HIS STAFF WORK WITH UPPER MANAGEMENT TO DEFINE A NEW PRODUCT. THIS DEFINITION IS DOCUMENTED IN THE FORM OF A SYSTEM CONFIGURATION INDEX DOCUMENT (CID) AND A SYSTEM REQUIREMENTS DOCUMENT. IT NORMALLY TAKES SEVERAL ITERATIONS TO PRODUCE THE AMOUNT OF DETAILED INFORMATION NEEDED TO PROCEED TO THE NEXT STEP.

THE NEXT STEP IS TO PARTITION THE FUNCTIONS THAT THIS SYSTEM WILL HAVE TO PERFORM INTO HARWARE AND SOFTWARE TASKS. AFTER THE FUNCTIONS THAT WILL BE IMPLEMENTED IN SOFTWARE HAVE BEEN IDENTIFIED, SOFTWARE REQUIREMENTS ARE DEFINED IN DETAIL AND THE SOFTWARE REQUIREMENTS DOCUMENT IS PREPARED. AT THIS TIME, THE PROJECT ENGINEERING MANAGER WILL HAVE ENOUGH INSIGHT INTO THE REQUIREMENTS OF THE PROJECT, AND ITS LEVEL OF EFFORT, THAT HE CAN ORGANIZE A SOFTWARE MANAGEMENT PLAN AND PREPARE THE SOFTWARE MANAGEMENT PLAN DOCUMENT.

THE SOFTWARE PART OF THE PROJECT HAS NOW REACHED A MAJOR PROJECT MILESTONE CALLED THE THE TOP-LEVEL SOFTWARE (OR STAGE-1) RELEASE. A PROJECT LEVEL DESIGN REVIEW SHOULD BE HELD TO CONSIDER DESIGN STATUS AND DESIGN ISSUES. THE DOCUMENTATION TO BE RELEASED SHOULD ALSO BE REVIEWED FOR ACCURACY AND COMPLETENESS. AFTER THE REVIEW IS COMPLETED, THE DOCUMENTS SHOULD BE CORRECTED AS NECESSARY AND A STAGE-1 RELEASE FORM COMPLETED AND SUBMITTED TO SOFTWARE CONFIGURATION MANAGEMENT. WE WILL NOW REVIEW THE ENTRIES ON THE ENCLOSED FORM LABELED:

STAGE-1 RELEASE FORM
TOP-LEVEL SOFTWARE

THE RELEASE AUTHORIZATION IS DATED AND SIGNED BY THE PROJECT ENGINEER AND SOFTWARE CONFIGURATION CONTROL MANAGER. COMPLETION OF THE RELEASE MILESTONE REQUIRES THAT ALL APPLICABLE INFORMATION BE COMPLETED ON THE FORM AND THAT THE RELEASED DOCUMENTS BE AVAILABLE IN THE PRINT ROOM. AFTER RELEASE, THE FORM IS FILED WITH CONFIGURATION CONTROL AS A RELEASE REFERENCE DOCUMENT.

THE FORM REQUIRES THE TOP-LEVEL SYSTEM NAME FOR REFERENCE PURPOSES. THIS IS NOT THE UNIT NAME BUT THE HIGHEST LEVEL SYSTEM NAME. IF THERE IS ONLY ONE LRU IN THE SYSTEM, THE SYSTEM NAME AND THE LRU NAME MAY BE THE SAME.

THE FORM REQUIRES THE REFERENCE DOCUMENT NUMBERS FOR THE FOLLOWING DOCUMENTS:

1. SYSTEM CID 700-XXXX-RN
2. SYSTEM REQUIREMENTS DOCUMENT 701-XXXX-RN

THESE DOCUMENTS ARE ALREADY RELEASED INTO THE SYSTEM. THEY ARE INCLUDED ONLY FOR UPWARD DOCUMENT REFERENCE.

THE MAJOR EMPHASIS OF THIS RELEASE IS THE SUBMITTAL OF THE FOLLOWING DOCUMENTS INTO THE DOCUMENT CONTROL SYSTEM:

1. SOFTWARE REQUIREMENTS DOCUMENT 705-XXXX-RN
2. SOFTWARE MANAGEMENT PLAN DOCUMENT 706-XXXX-RN

THESE ARE SUBMITTED AND THEIR NUMBERS ARE ASSIGNED IN THE SAME MANNER AS OTHER DOCUMENTS ARE CURRENTLY SUBMITTED. THE NUMBERS MAY BE PREASSIGNED WHEN NEEDED FOR REFERENCE PURPOSES.

NEXT, DOCUMENT NUMBERS ARE PREASSIGNED FOR THE FOLLOWING DOCUMENTS:

1. LRU SOFTWARE DOCUMENT 715-XXXX-RN
2. LRU SOFTWARE STRUCTURE DIAGRAM 716-XXXX-RN
3. LRU SOFTWARE TEST DOCUMENT 707-XXXX-RN

THERE ARE A SET OF THESE DOCUMENT NUMBERS REQUIRED FOR EVERY NEW LRU (UNIT OR 'BOX') IN THE SYSTEM. THE FORM ASKS FOR THE UNIT NAME AND THE HARDWARE PART NUMBER FOR REFERENCE PURPOSES. THERE IS ROOM ON THE STANDARD FORM FOR FIVE NEW UNITS. ADD A SECOND SHEET IF MORE SPACES ARE REQUIRED.

THE BOTTOM PART OF THE FORM CONTAINS THE DISTRIBUTION FOR THE RELEASE FORM AFTER THE RELEASE IS COMPLETED. NO SIGNATURES ARE REQUIRED.

THE SECOND MAJOR SOFTWARE PROJECT MILESTONE IS CALLED THE LRU (UNIT) SOFTWARE DOCUMENTATION (OR STAGE-2) RELEASE. IT RELEASES THE TOP LEVEL DOCUMENTS FOR EACH NEW LRU (UNIT) IN THE SYSTEM. THE LRU SOFTWARE DOCUMENT (715-XXXX-RN) IS THE MAIN SOFTWARE REFERENCE FOR ALL THE SOFTWARE CONTAINED IN THE LRU. A STAGE-2 RELEASE FORM IS REQUIRED FOR EACH LRU. ANOTHER MAJOR PROJECT-LEVEL DESIGN REVIEW SHOULD BE HELD AT THIS TIME. THE STAGE-2 RELEASE WILL USUALLY OCCUR ABOUT TWO-THIRDS OF THE WAY THROUGH THE DESIGN AND DEVELOPMENT PHASE OF THE PROJECT. MOST OF THE DESIGN ISSUES HAVE BEEN DECIDED AND A MAJOR PART OF THE BASELINE DEVELOPMENT WILL HAVE BEEN COMPLETED. THE ADDITIONAL DOCUMENTATION THAT WILL BE RELEASED SHOULD AGAIN BE REVIEWED FOR ACCURACY AND COMPLETENESS AND CORRECTED AS NEEDED. A STAGE-2 RELEASE FORM IS THEN COMPLETED AND SUBMITTED TO SOFTWARE CONFIGURATION MANAGEMENT.

WE WILL NOW REVIEW THE ENTRIES ON THE ENCLOSED FORM LABELED:

STAGE-2 RELEASE FORM
LRU SOFTWARE DOCUMENTATION

AS WITH THE PREVIOUS RELEASE, THE AUTHORIZATION IS DATED AND SIGNED BY THE PROJECT ENGINEER AND SOFTWARE CONFIGURATION CONTROL MANAGER. COMPLETION OF THE RELEASE MILESTONE REQUIRES THAT ALL APPLICABLE INFORMATION BE COMPLETED ON THE FORM AND THAT THE RELEASED DOCUMENTS BE AVAILABLE IN THE PRINT ROOM. AFTER RELEASE, THE FORM IS FILED WITH CONFIGURATION CONTROL AS A RELEASE REFERENCE DOCUMENT.

THE FORM REQUIRES THE FOLLOWING INFORMATION FOR UPWARD REFERENCE PURPOSES:

1. THE TOP-LEVEL SYSTEM NAME.
2. THE LRU (UNIT) NAME.
3. THE LRU HARDWARE PART NUMBER.
4. THE LRU SOFTWARE DOCUMENT NUMBER. 715-XXXX-RN

THE MAJOR EMPHASIS OF THE STAGE-2 SOFTWARE RELEASE IS THE SUBMITTAL OF THE FOLLOWING DOCUMENTS INTO THE DOCUMENT CONTROL SYSTEM:

1. LRU SOFTWARE DOCUMENT 715-XXXX-RN
2. LRU SOFTWARE STRUCTURE DIAGRAM 716-XXXX-RN
3. LRU SOFTWARE TEST DOCUMENT 707-XXXX-RN

ALL OF THE ABOVE SOFTWARE NUMBERS WERE PREASSIGNED IN THE STAGE-1 RELEASE SO NO NEW NUMBERS ARE NEEDED.

IN THE STAGE-2 RELEASE, DOCUMENT NUMBERS ARE ASSIGNED TO THE DOCUMENTS THAT SUPPORT EACH MICROPROCESSOR PROGRAM IN THE DESIGNATED LRU. FOR EACH PROCESSOR IN THE LRU, THE FOLLOWING DOCUMENT NUMBERS ARE PREASSIGNED:

1.	LOAD FILE RELEASE DOCUMENT	720-XXXX-RN
2.	DESIGN DESCRIPTION DOCUMENT	721-XXXX-RN
3.	SOURCE LISTING DOCUMENT	719-XXXX-RN
4.	PROGRAMMER'S REFERENCE LITERATURE	718-XXXX-RN

THE KEY REFERENCE DOCUMENT IS THE LOAD FILE RELEASE. IT WILL PROVIDE ALL OF THE REFERENCE INFORMATION NEEDED TO RECREATE THE PROCESSOR CODE OR MODIFY IT. REMEMBER THAT A SET OF FOUR DOCUMENT NUMBERS MUST BE ASSIGNED FOR EACH PROCESSOR IN THE LRU. THEIR IS ROOM ON THE STANDARD FORM FOR FIVE PROCESSORS IN A SINGLE LRU. IF MORE PROCESSORS ARE USED, ADD MORE SHEETS AS REQUIRED.

THE BOTTOM PART OF THE FORM CONTAINS THE DISTRIBUTION FOR THE RELEASE FORM AFTER THE RELEASE IS COMPLETED. NO SIGNATURES ARE REQUIRED.

AFTER THE STAGE-2 RELEASE IS COMPLETED BUT BEFORE THE PURCHASING BILL-OF-MATERIALS IS RELEASED, YOU SHOULD APPLY TO COMPONENT ENGINEERING FOR THE RELEASE OF A SPECIFICATION CONTROL DRAWING FOR EACH PROGRAMMED PROCESSOR OR MEMORY CHIP IN EACH NEW LRU IN THE SYSTEM. THE PRINCIPAL REASON FOR DOING THIS IS TO OBTAIN A PART NUMBER THAT CAN BE PUT ON THE BILL-OF-MATERIALS. THIS PART NUMBER AND ITS ASSOCIATED SPECIFICATION CONTROL DRAWING WILL BE USED BY PURCHASING PLANNERS TO SCHEDULE THE DELIVERY OF ELECTICAL PARTS. THIS RELEASE REQUIRES THREE STEPS:

1. APPLY TO COMPONENT ENGINEERING FOR A 122-XXXX-RN NUMBER ASSIGNMENT.
2. APPLY TO DOCUMENT CONTROL FOR A 722-XXXX-RN DOCUMENT NUMBER FOR THE BINARY IMAGE FILE THAT WILL BE PROGRAMMED INTO THE CHIP.
3. WHEN THE BINARY IMAGE FILE DOCUMENT NUMBER HAS BEEN ASSIGNED, TAKE A COPY OF THE FORM TO COMPONENT ENGINEERING AND REQUEST THAT THE 122-XXXX-RN SPECIFICATION CONTROL DRAWING THAT APPLIES TO THAT PROGRAMMED CHIP BE RELEASED INTO THE DOCUMENT CONTROL SYSTEM.

AT THIS POINT, THE PROGRAMMED I.C. SPECIFICATION CONTROL DRAWING WILL BE AVAILABLE TO THE PURCHASING PLANNERS BUT THE SOFTWARE IS NOT YET RELEASED SO THAT THE CHIPS CAN BE PROGRAMMED.

LOAD FILE SOFTWARE RELEASE

THE LAST MAJOR SOFTWARE RELEASE OCCURS NEAR THE END OF THE DESIGN AND DEVELOPMENT CYCLE AND JUST BEFORE THE MANUFACTURING RELEASE. THIS RELEASE (STAGE-3) IS CALLED THE LOAD FILE RELEASE BECAUSE IT INITIATES THE PROCESS OF CREATING THE FIRST VERSION OF LOADABLE SOFTWARE THAT IS PUT UNDER REVISION CONTROL. THE LAST OF THE SUPPORT DOCUMENTATION IS ALSO RELEASED AT THIS TIME. A STAGE-3 RELEASE FORM IS REQUIRED FOR EACH PROCESSOR IN THE NEW SYSTEM. IF A PROCESSOR CONSISTS OF MANY PROGRAMMED PARTS, THESE ARE ALL HANDLED WITH A SINGLE FORM. A FINAL MAJOR PROJECT-LEVEL REVIEW OF THE SOFTWARE SHOULD BE HELD PRIOR TO RELEASE. THIS REVIEW IS MAINLY CONCERNED WITH INSURING THAT ALL FUNCTIONAL REQUIREMENTS CALLED OUT IN THE SOFTWARE REQUIREMENTS DOCUMENT HAVE BEEN MET. THE NEW DOCUMENTATION THAT WILL BE RELEASED SHOULD BE REVIEWED FOR ACCURACY AND COMPLETENESS AND CORRECTED AS NEEDED. ALSO, ALL PREVIOUSLY RELEASED DOCUMENTS SHOULD BE REVIEWED FOR ACCURACY AND COMPLETENESS AND UPDATED AS NEEDED. AFTER THESE UPDATES ARE FINISHED, A STAGE-3 RELEASE FORM IS SUBMITTED TO SOFTWARE CONFIGURATION MANAGEMENT. AT THIS POINT, THE SOFTWARE DESIGN AND DEVELOPMENT CYCLE IS COMPLETED AND THE RELEASED SOFTWARE IS MATURE ENOUGH TO GO INTO THE MAINTAINANCE STAGE OF ITS LIFE-CYCLE.

WE WILL NOW REVIEW THE ENTRIES ON THE ENCLOSED FORM LABELED:

STAGE-3 RELEASE FORM
LOAD FILE SOFTWARE

THE FIRST PART OF THE FORM DIFFERS FROM PREVIOUS RELEASES IN THAT THE PROJECT ENGINEER IS REQUIRED TO SIGN THAT HE HAS VERIFIED THE LOAD FILE PRODUCED WITH THE SUPPORT SOFTWARE COMMAND FILE. THIS VERIFICATION SHOULD INCLUDE BOTH CHIP CONTENTS AND FUNCTIONAL PERFORMANCE. AFTER VERIFICATION, THE SOFTWARE CONFIGURATION CONTROL MANAGER SIGNS THE RELEASE.

COMPLETION OF THIS RELEASE MILESTONE REQUIRES THAT THE ENTIRE

DOCUMENTATION AND TARGET SOFTWARE PACKAGE FOR THE SYSTEM AND PROCESSOR BE COMPLETED AND ALL RELEASED DOCUMENTS BE AVAILABLE IN THE PRINT ROOM. AFTER RELEASE, THE FORM IS FILED WITH CONFIGURATION CONTROL AS A RELEASE REFERENCE DOCUMENT.

THE LOAD FILE RELEASE IS DIFFERENT FROM THE OTHERS IN THAT IT RELEASES THE LOADABLE OBJECT FILE SO PARTS THAT ARE SPECIFIED IN A 122-XXXX-RN SPECIFICATION CONTROL DRAWING CAN BE PROGRAMMED FOR PRODUCTION. THE MAJOR GOAL OF THE LOAD FILE RELEASE IS TO PROVIDE COMPLETE DOCUMENTATION OF ALL TARGET SOFTWARE AND SUPPORT SOFTWARE AND PROVIDE THE NECESSARY INFORMATION TO RECREATE THE LOAD FILE AND ITS VARIOUS FUTURE REVISIONS AT ANY TIME DURING THE LIFE CYCLE OF THE PRODUCT.

THE FIRST ITEM IS THE SUPPORT SOFTWARE COMMAND FILE. THIS IS A FILE WRITTEN BY THE SOFTWARE ENGINEER THAT CAN BE RUN BY SOFTWARE CONTROL TO PRODUCE THE TARGET LOAD FILE. THIS COMMAND FILE MUST BE COMPLETE IN EVERY DETAIL AS IT BECOMES PART OF THE DOCUMENTATION PACKAGE ON HOW TO CREATE THE LOAD FILE. THE COMMAND FILE SHOULD RUN WITHOUT ANY PROMPTS OR INTERVENTION BY THE SOFTWARE CONTROL LIBRARIAN.

NEXT IS THE VAX FILE NAME FOR THE LOAD FILE. THIS IS CHOSEN BY THE SOFTWARE ENGINEER AS A DESCRIPTIVE NAME FOR THE TARGET SOFTWARE TASK.

THE ARCHIVE VOLUME NAME AND DATE IS SUPPLIED BY THE SOFTWARE CONTROL LIBRARIAN. ALL OF THE TARGET SOFTWARE WILL BE ARCHIVED IN AN ARCHIVE CONTAINER FILE NAME CORRESPONDING TO THE LOAD FILE DOCUMENT NUMBER (720XXXXRN). THIS IS DONE AT THE SAME TIME THAT THE LOAD FILE IS CREATED.

FOR MOST FUTURE PROJECTS, THE LOAD FILE CREATED BY THE LINKING LOADER WILL BE LARGER THAN CAN BE PROGRAMMED INTO A SINGLE CHIP. ALSO, THE LOAD FILE FOR 16-BIT PROCESSORS WILL, MOST LIKELY, HAVE TO BE LOADED INTO BYTE-ORGANIZED MEMORY. TO TAKE CARE OF THESE FACTORS, MORE SUPPORT SOFTWARE IS AVAILABLE TO SEPARATE THE SINGLE LOAD FILE INTO ODD AND EVEN BYTES, PARTITION THE LOADABLE OBJECT CODE INTO EACH PHYSICAL MEMORY DEVICE, AND FORMAT AN ASCII-HEX FILE FOR EACH MEMORY DEVICE SO ITS CORRESPONDING CODE CAN BE DOWNLOADED TO A PROM PROGRAMMER OR INTEGRATION UNIT.

THE CODE IN EACH MEMORY DEVICE WILL BE DOCUMENTED BY A BINARY IMAGE FILE. THIS DOCUMENT IS A LISTING OF THE MEMORY LOCATIONS AND OP CODE APPLICABLE TO EACH PROGRAMMED DEVICE. EACH BINARY IMAGE FILE HAS A 722-XXXX-RN DOCUMENT NUMBER AND A DOCUMENT FILE NAME ON VAX. THIS DOCUMENT IS A PRINTOUT OF THE CODE IN THE DEVICE. ANOTHER SUPPORT SOFTWARE PACKAGE USES THIS DOCUMENTED CODE TO CREATE THE ACTUAL ASCII-HEX FILE THAT WILL BE USED TO PROGRAM EACH DEVICE. THE ASCII-HEX FILE NAME WILL BE THE NINE DIGIT DOCUMENT NUMBER FOR THE BINARY IMAGE FILE WITH A FILE EXTENSION THAT CORRESPONDS TO THE TARGET CPU.

SINCE THE LOAD FILE IS ALSO AVAILABLE IN ASCII-HEX FORMAT, IT MAY CAUSE SOME CONFUSION AS TO WHY THE ADDITIONAL FILES ARE CREATED. THE BINARY IMAGE FILE IS USED AS A HARD-COPY DOCUMENTATION OF THE CODE IN EACH MEMORY DEVICE. AS AN ADDITIONAL CHECK OF THE ACCURACY OF THIS DOCUMENT, THE BINARY IMAGE FILE IS ACTUALLY USED TO RE-CREATE A DOWNLOADABLE ASCII-HEX FILE. IT IS THIS FILE THAT IS USED TO PROGRAM THE MASTER DEVICES THAT WILL BE USED IN PRODUCTION.

THE FORM REQUIRES THE LISTING OF ALL TARGET SOURCE CODE FILE NAMES AND THEIR CORRESPONDING RELOCATABLE OBJECT FILE NAMES. THE SOURCE FILE NAMES MUST AGREE WITH THE FILE NAMES CALLED OUT IN THE COMMAND FILE USED TO CREATE THE LOAD FILE. THE OBJECT FILE NAMES ARE DOCUMENTED AS REQUIRED BY DD-178.

THE SUPPORT SOFTWARE USED BY THE SOFTWARE DEVELOPER TO CREATE THE FINISHED VERSION OF THE LOAD FILE MUST ALSO BE DOCUMENTED. EXACT FILE NAMES AND DATES MUST BE GIVEN AND MUST AGREE WITH THE SUPPORT SOFTWARE COMMAND FILE NAMES. THE FILE NAMES THAT APPLY FOR EACH CATAGORY OF SUPPORT SOFTWARE CAN BE OBTAINED FROM SUPPORT SOFTWARE PERSONNEL. THOSE ITEMS OF SUPPORT SOFTWARE NOT USED SHOULD BE MARKED N/U.

THE FORM REQUIRES THE FOLLOWING INFORMATION FOR UPWARD REFERENCE PURPOSES:

1. THE TOP-LEVEL SYSTEM NAME.
2. THE LRU (UNIT) NAME.
3. THE LRU SOFTWARE DOCUMENT NUMBER. 715-XXXX-RN
4. THE DESIGN DESCRIPTION DOCUMENT NUMBER 721-XXXX-RN
5. THE SOURCE LISTING DOCUMENT NUMBER 719-XXXX-RN
6. THE PROGRAMMER'S REFERENCE LITERATURE DOCUMENT NUMBER 718-XXXX-RN

THE BOTTOM PART OF THE FORM CONTAINS THE DISTRIBUTION FOR THE RELEASE

FORM AFTER THE RELEASE IS COMPLETED. NO SIGNATURES ARE REQUIRED.

BEFORE THE STAGE-3 RELEASE IS ACTUALLY COMPLETED AND SIGNED OFF, MASTER COPIES OF ALL PROGRAMMED CHIPS NEED TO BE CREATED AND VERIFIED. FOR THE PRESENT TIME, THE SOFTWARE LIBRARIAN AND THE SOFTWARE PROJECT ENGINEER WILL WORK TOGETHER TO CREATE THE LOAD FILE AND BINARY IMAGE FILES ON VAX AND PROGRAM FOUR (4) MASTER COPIES OF EACH UV PROM. THESE FOUR CHIP SETS WILL BE LABELED WITH THEIR CORRESPONDING 122-XXXX-RN PART NUMBERS.

THE PROJECT ENGINEER WILL TAKE ONE OF THE CHIP SETS AND USE IT FOR VERIFICATION AGAINST THE CORRESPONDING CHIP THAT HE USES IN THE TESTED ENGINEERING MODEL. AFTER DETAILED TESTING, THE SOFTWARE PROJECT ENGINEER WILL BE REQUIRED TO SIGN OFF ON THE RELEASE FORM THAT HE HAS COMPLETED THIS VERIFICATION PROCESS.

AFTER THE VERIFICATION PROCESS HAS SHOWN THAT THE CHIPS CONTAIN THE CORRECT SOFTWARE, THE THREE REMAINING MASTER COPIES ARE SUBMITTED TO COMPONENT ENGINEERING FOR SAFEKEEPING. ONE OF THESE COPIES WILL BE KEPT AS A PERMENANT RECORD AND THE OTHER TWO COPIES WILL BE ON ON HAND WHEN RECEIVING INSPECTION NEEDS THEM.

QUANTITY PROGRAMMING OF UV PROM CHIPS FOR PRODUCTION IS ACCOMPLISHED AS FOLLOWS:

1. RECEIVING INSPECTION (R.I.) RECEIVES A WORK ORDER FROM THE PURCHASING PLANNER. THE WORK ORDER REQUESTS A QUANTITY OF PARTS TO BE WORKED UP PER A 122-XXXX-RN DRAWING. THE PLANNER ALSO PROVIDES R.I. WITH THE UNPROGRAMMED 120-XXXX-XX PARTS.

2. R.I. REQUESTS TWO COPIES OF THE MASTER CHIP FROM COMPONENT ENGINEERING. ONE OF THESE WILL BE USED AS THE SOURCE FOR GANG PROGRAMMING THE PRODUCTION DEVICES. IT WILL BE THE SO-CALLED 'WORKING COPY'. THE OTHER WILL BE MAINTAINED IN A PRISTINE CONDITION SO THAT IT MAY BE USED FOR QUALITY VERIFICATION OF THE GANG PROGRAMMING PROCESS. PROGRAMMED PARTS ARE THEN STOCKED UNDER THE 122-XXXX-RN PART NUMBER THAT APPEARS ON THE B/M.

NOTICE THAT ONCE THE PROJECT ENGINEER HAS HELPED SOFTWARE CONTROL TO PRODUCE THE FOUR MASTER COPIES, ENGINEERING IS NO LONGER INVOLVED IN THE PROGRAMMING OF CHIPS. COMPONENT ENGINEERING ALSO HAS EXTRA COPIES OF THE MASTER CHIP AVAILABLE SO THAT THE GANG PROGRAMMED PARTS CAN BE VERIFIED BY A CHIP THAT WAS NOT USED TO PROGRAM THEM BUT IS OTHERWISE IDENTICAL.

WHEN REVISIONS ARE MADE TO THE SOURCE CODE, THE LOAD FILE RELEASE PROCEDURE IS REPEATED AND THE DASH NUMBER ON AFFECTED SOFTWARE DOCUMENTS AND PART NUMBERS IS INCREMENTED.

DOCUMENT DEFINITIONS

SYSTEM CONFIGURATION INDEX DOCUMENT (SYSTEM CID)
700-XXXX-RN

THE SYSTEM CID IS A TOP-LEVEL LISTING FOR THE SYSTEM CONTAINING HARDWARE NAMES AND PART NUMBERS AND SOFTWARE DOCUMENT NUMBERS FOR ALL OF THE LINE REPLACEABLE UNITS (LRUS). IT IS ESSENTIALLY A BOX-LEVEL DOCUMENTATION OF ALL OF THE UNITS REQUIRED IN THE SYSTEM. THE DOCUMENTS THAT ARE CURRENTLY USED IN KING RADIO THAT ARE SIMILIAR TO THE SYSTEM CID ARE THE PRODUCT STRUCTURE DIAGRAMS USED IN ENGINEERING AND THE CERTIFICATION DOCUMENT PACKAGES PRODUCED BY RALPH COLE'S GROUP. THE PRODUCT STRUCTURE DIAGRAM IS NOT EXACTLY THE SAME AS THE SYSTEM CID BECAUSE OF THE MANY SYSTEM FLAVORS THAT ARE SOMETIMES POSSIBLE. EACH PROJECT ENGINEERING MANAGER WILL HAVE TO EVALUATE HIS PROJECT TO DETERMINE WHICH FORM OF DOCUMENT BEST FITS HIS REQUIREMENT FOR THE SYSTEM CID.

SYSTEM DEFINITION DOCUMENT
701-XXXX-RN

THE SYSTEM DEFINITION DOCUMENT IS A HIGH LEVEL DESCRIPTION OF THE DEFINITION OF THE PRODUCT. IT CONTAINS BLOCK DIAGRAMS OF THE SYSTEM AND DESCRIPTIONS OF THE FUNCTIONS TO BE PERFORMED BY EACH. IT DESCRIBES THE OVERALL PERFORMANCE REQUIREMENTS AND HUMAN FACTORS CONSIDERATIONS. IT CONTAINS A FURTHER BREAKDOWN OF DESIRED FUNCTIONS TO SUCH A LEVEL OF DETAIL THAT LATER PARTITIONING OF FUNCTIONS INTO HARDWARE AND SOFTWARE IMPLEMENTATIONS CAN BE MET.

COPIES: GARY BURRELL — ALL GROUP LEADERS
DALE COOPER — SOFTWARE ENGINEERS AND PROGRAMMERS
TOM DENNIS — RALPH COLE

** 6.0 INSTRUCTIONS FOR FILLING OUT RELEASE FORMS

** 6.1 INTRODUCTION

TELL ABOUT ORGAINZATION OF THE FORM. UPWARD REFERECE, RELEASE, AND PREASSIGNMENT OF NUMBERS.

** 6.2 STAGE-1 RELEASE FORM (799-0001-00).

ARE:

1) UPWARD REFERENCE TO THE SYSTEM CID AND SYSTEM SPECIFICATION.
2) RELEASE TO DOCUMENT CONTROL OF THE SOFTWARE REQUIREMENTS AND SOFTWARE MANAGEMENT PLAN DOCUMENTS.
3) PREASSIGNMENT OF DOCUMENT NUMBERS FOR THE LRU LEVEL DOCUMENTS.

THE FIRST TWO LINES OF THE FORM ARE THE RELEASE DATE AND THE SIGNATURES OF THE PROJECT ENGINEER AND SOFTWARE CONFIGURATION CONTROL MANAGER. COMPLETION OF THE RELEASE MILESTONE REQUIRES THAT ALL APPLICABLE INFORMATION BE COMPLETED ON THE FORM AND THAT THE RELEASED DOCUMENTS BE AVAILABLE IN THE PRINT ROOM. WHEN THESE ARE DONE, THE FORM IS SIGNED AND FILED WITH CONFIGURATION CONTROL AS A RELEASE REFERENCE DOCUMENT.

NEXT IS THE TOP-LEVEL SYSTEM NAME AS REQUIRED FOR REFERENCE PURPOSES. THIS IS NOT THE UNIT NAME BUT THE HIGHEST LEVEL SYSTEM NAME. IF THERE IS ONLY ONE LRU IN THE SYSTEM, THE SYSTEM NAME AND THE LRU NAME COULD BE THE SAME.

DOCUMENT NUMBERS, FOR UPWARD REFERENCE PURPOSES, ARE REQUIRED FOR THE FOLLOWING DOCUMENTS:

1. SYSTEM CID — 700-XXXX-RN
2. SYSTEM REQUIREMENTS DOCUMENT — 701-XXXX-RN

THESE DOCUMENTS ARE ALREADY RELEASED INTO THE SYSTEM.

THE MAJOR EMPHASIS OF THIS RELEASE IS THE SUBMITTAL OF THE FOLLOWING SYSTEM LEVEL SOFTWARE DOCUMENTS INTO THE DOCUMENT CONTROL SYSTEM:

1. SOFTWARE REQUIREMENTS DOCUMENT — 705-XXXX-RN
2. SOFTWARE MANAGEMENT PLAN DOCUMENT — 706-XXXX-RN

THESE ARE SUBMITTED AND THEIR NUMBERS ARE ASSIGNED IN THE SAME MANNER AS OTHER DOCUMENTS ARE CURRENTLY SUBMITTED. THESE NUMBERS ARE NORMALLY ASSIGNED AT THE TIME OF THE STAGE-1 RELEASE BUT MAY BE PREASSIGNED WHEN NEEDED FOR REFERENCE PURPOSES.

FINALLY, NUMBERS ARE PREASSIGNED FOR THE FOLLOWING LRU LEVEL DOCUMENTS:

1. LRU SOFTWARE DOCUMENT — 715-XXXX-RN
2. LRU SOFTWARE STRUCTURE DIAGRAM — 716-XXXX-RN
3. LRU SOFTWARE TEST DOCUMENT — 707-XXXX-RN

THERE ARE A SET OF THESE DOCUMENT NUMBERS REQUIRED FOR EVERY NEW LRU (UNIT OR '80X') IN THE SYSTEM. FOR CROSS-REFERENCE PURPOSES, THE FORM ALSO REQUIRES THE UNIT NAME AND THE HARDWARE PART NUMBER. THERE IS ROOM ON THE STANDARD FORM FOR FIVE NEW UNITS. ADD A SECOND SHEET IF MORE SPACES ARE REQUIRED. REMEMBER, THESE DOCUMENTS ARE NOT RELEASED AT THIS TIME BUT THEIR NUMBERS ARE PREASSIGNED TO PROVIDE REFERENCE NUMBERS, IF NEEDED, AND TO EMPHASIZE THAT THEY WILL BE RELEASED AT THE NEXT STAGE.

THE BOTTOM OF THE FORM CONTAINS THE DISTRIBUTION LIST FOR THE RELEASE FORM AFTER THE RELEASE IS COMPLETED. THE DATE IS THE ACTUAL DATE THAT COPIES OF THE RELEASE FORM ARE DISTRIBUTED.

** 6.3 STAGE-2 RELEASE FORM (799-0002-00)

THE MAIN OBJECTIVES OF THE STAGE-2 RELEASE ARE:

1) UPWARD REFERENCE TO SYSTEM-LEVEL DOCUMENTS AND NAMES.
2) RELEASE TO DOCUMENT CONTROL OF THE LRU-LEVEL DOCUMENTS.
3) PREASSIGNMENT OF DOCUMENT NUMBERS FOR THE PROCESSOR-LEVEL DOCUMENTS.

A STAGE-2 RELEASE IS REQUIRED FOR EACH NEW LRU IN THE SYSTEM.

AS WITH THE PREVIOUS FORM, THE FIRST TWO LINES ARE THE RELEASE DATE AND THE SIGNATURES OF THE PROJECT ENGINEER AND SOFTWARE CONFIGURATION CONTROL MANAGER. COMPLETION OF THE RELEASE MILESTONE REQUIRES THAT ALL APPLICABLE INFORMATION BE COMPLETED ON THE FORM AND THAT THE RELEASED DOCUMENTS BE AVAILABLE IN THE PRINT ROOM. WHEN THESE ARE DONE, THE FORM IS SIGNED AND FILED WITH CONFIGURATION CONTROL AS A RELEASE REFERENCE DOCUMENT.

THE FORM REQUIRES THE FOLLOWING INFORMATION FOR UPWARD REFERENCE PURPOSES:

1. THE TOP-LEVEL SYSTEM NAME.
2. THE LRU (UNIT) NAME.
3. THE LRU HARDWARE PART NUMBER. XXX-XXXX-XX
4. THE LRU SOFTWARE DOCUMENT NUMBER. 715-XXXX-RN

DOCUMENT NUMBERS, FOR UPWARD REFERENCE PURPOSES, ARE REQUIRED FOR THE FOLLOWING:

1. SYSTEM CID 700-XXXX-RN
2. SYSTEM SPECIFICATION 701-XXXX-RN
3. SOFTWARE REQUIREMENTS 705-XXXX-RN
4. SOFTWARE MANAGEMENT PLAN 706-XXXX-RN

THESE DOCUMENTS ARE ALREADY RELEASED INTO THE SYSTEM.

THE MAJOR EMPHASIS OF THE STAGE-2 SOFTWARE RELEASE IS THE SUBMITTAL OF THE FOLLOWING DOCUMENTS INTO THE DOCUMENT CONTROL SYSTEM:

1. LRU SOFTWARE 715-XXXX-RN
2. LRU SOFTWARE STRUCTURE DIAGRAM 716-XXXX-RN
3. LRU SOFTWARE TEST 707-XXXX-RN

ALL OF THE ABOVE SOFTWARE NUMBERS WERE PREASSIGNED IN THE STAGE-1 RELEASE SO NO NEW NUMBERS ARE NEEDED.

IN THE STAGE-2 RELEASE, THE FOLLOWING NUMBERS ARE PRE-ASSIGNED FOR THE DOCUMENTS THAT SUPPORT EACH MICROPROCESSOR PROGRAM IN THE DESIGNATED LRU:

1. LOAD FILE RELEASE 720-XXXX-RN
2. DESIGN DESCRIPTION 721-XXXX-RN
3. SOURCE LISTING 719-XXXX-RN
4. PROGRAMMER'S REFERENCE LITERATURE 718-XXXX-RN

THE KEY REFERENCE DOCUMENT IS THE LOAD FILE RELEASE. IT WILL PROVIDE ALL OF THE REFERENCE INFORMATION NEEDED TO RECREATE THE PROCESSOR CODE OR MODIFY IT. REMEMBER THAT A SET OF FOUR DOCUMENT NUMBERS MUST BE ASSIGNED FOR EACH PROCESSOR IN THE LRU. THEIR IS ROOM ON THE STANDARD FORM FOR FIVE PROCESSORS IN A SINGLE LRU. IF MORE PROCESSORS ARE USED, ADD MORE SHEETS AS REQUIRED.

THE BOTTOM OF THE FORM CONTAINS THE DISTRIBUTION LIST FOR THE RELEASE FORM AFTER THE RELEASE IS COMPLETED. THE DATE IS THE ACTUAL DATE THAT COPIES OF THE RELEASE FORM ARE DISTRIBUTED.

** 6.4 STAGE-3 RELEASE FORM (799-0003-00).

THE STAGE-3 RELEASE FORM IS DIFFERENT FROM THE OTHERS IN THE FOLLOWING AREAS:

1) IT REQUIRES A LOAD FILE SOFTWARE DOCUMENT NUMBER BECAUSE ALL OF THE INFORMATION ON THE FORM WILL BE RELEASED AS A CONTROLLED DOCUMENT.
2) IT RELEASES THE LOADABLE OBJECT FILES SO PARTS THAT ARE SPECIFIED IN A 122-XXXX-RN SPECIFICATION CONTROL DRAWING CAN BE PROGRAMMED FOR PRODUCTION.
3) THE PROJECT ENGINEER IS REQUIRED TO SIGN THE FORM TESTIFYING THAT HE AS VERIFIED ALL OF THE MASTER CHIPS PROGRAMMED FROM THE ASCII-HEX FILES THAT WERE PRODUCED USING THE SUPPORT SOFTWARE COMMAND PROCEDURE.
4) REFERENCE IS MADE TO THE SUPPORT SOFTWARE USED TO CREATE THE LOADABLE FILES.

THE PRIMARY GOAL OF THIS FORM IS TO COLLECT, INTO A SINGLE DOCUMENT, ALL OF THE INFORMATION, THAT IS REQUIRED TO CREATE THE LOAD FILE, ODD AND EVEN BYTE FILES (IF NEEDED), BINARY IMAGE DOCUMENT FILES, AND ASCII-HEX FILES. THIS INFORMATION WILL BE USED TO RECREATE THE LOAD FILE AND SUPPORTING FILES AT ANY TIME MAINTAINENCE IS NEEDED DURING THE LIFE CYCLE OF THE PRODUCT.

COMPLETION OF THIS RELEASE MILESTONE REQUIRES THAT THE ENTIRE DOCUMENTATION AND TARGET SOFTWARE PACKAGE FOR THE SYSTEM AND PROCESSOR BE COMPLETED AND ALL DOCUMENTS BE AVAILABLE IN THE PRINT ROOM. IT ALSO REQUIRES THAT VERIFICATION OF THE MASTER CHIPS BE COMPLETED AND THAT THEY ARE AVAILABLE FROM COMPONENT ENGINEERING. WHEN THESE ARE DONE, THE FORM IS SIGNED BY SOFTWARE CONTROL AND FILED AS A RELEASE REFERENCE DOCUMENT.

THE FIRST ITEM ON THE FORM IS THE LOAD FILE RELEASE DOCUMENT NUMBER. THIS WAS PREASSIGNED IN THE STAGE-2 RELEASE.

NEXT IS THE PROCESSOR SYSTEM NAME. THIS SHOULD BE A DESCRIPTIVE NAME DENOTING THE FUNCTION OF THE PROCESSOR IN THE LRU. THE NEXT LINE IS THE RELEASE AUTHORIZATION FOR THE LOAD FILE. IT IS SIGNED BY SOFTWARE CONTROL AFTER ALL OTHER TASKS ARE COMPLETED.

THE VERIFICATION LINE IS DATED AND SIGNED BY THE PROJECT ENGINEER AFTER ALL PROGRAMMED MEMORY DEVICES CREATED BY THE SUPPORT SOFTWARE COMMAND PROCEDURE HAVE BE VERIFIED.

NEXT IS THE SUPPORT SOFTWARE COMMAND FILE NAME. THIS IS A FILE WRITTEN BY THE SOFTWARE ENGINEER THAT CAN BE RUN BY SOFTWARE CONTROL TO PRODUCE THE TARGET LOAD FILE. THIS COMMAND FILE MUST BE COMPLETE IN EVERY DETAIL AS IT BECOMES PART OF THE DOCUMENTATION PACKAGE ON HOW TO CREATE THE LOAD FILE. THE COMMAND FILE SHOULD RUN WITHOUT ANY PROMPTS OR INTERVENTION BY THE SOFTWARE CONTROL LIBRARIAN.

NEXT IS THE VAX FILE NAME FOR THE LOAD FILE. THIS IS CHOSEN BY THE SOFTWARE ENGINEER AS A DESCRIPTIVE NAME FOR THE TARGET SOFTWARE TASK.

THE ARCHIVE VOLUME NAME AND DATE IS SUPPLIED BY THE SOFTWARE CONTROL LIBRARIAN. ALL OF THE TARGET SOFTWARE WILL BE ARCHIVED IN AN ARCHIVE CONTAINER FILE NAME CORRESPONDING TO THE LOAD FILE DOCUMENT NUMBER (720XXXXRN). THIS IS DONE AFTER THE MASTER CHIPS HAVE BEEN VERIFIED BY THE PROJECT ENGINEER AND JUST BEFORE THE LOAD FILE DOCUMENT IS RELEASED.

THE NEXT SEVERAL LINES CONTAIN DOCUMENTATION REFERENCES AT THE PROGRAMMED MEMORY DEVICE LEVEL. A GENERIC CLASSIFICATION CALLED BINARY IMAGE IS USED BECAUSE THESE FILES REPRESENT VARIOUS FORMATS OF THE ACTUAL BINARY CODE IN THE PROGRAMMED DEVICE.

THE FIRST TYPES OF FILES ARE THE EVEN/ODD BYTE FILES. THESE ARE NEEDED WHEN THE LOAD FILE FOR A 16-BIT PROCESSOR MUST BE LOADED INTO BYTE-ORGANIZED MEMORY. A SUPPORT SOFTWARE PROGRAM CALLED THE SEPARATOR IS USED TO SORT THE ODD AND EVEN BYTES AND SEPARATE THEM INTO TWO FILES HAVING THE SAME FILE NAME AS THE LOAD FILE BUT DIFFERENTIATED BY UNIQUE FILE EXTENSIONS. FOR 8-BIT PROCESSORS, THIS BLANK IS LABELED N/U BECAUSE THIS STEP IS NOT USED. THESE FILES ARE SAVED ON VAX AND ARCHIVED UNDER THE LOAD FILE SAVE-SET NAME BUT ARE NOT PUT INTO A RELEASED DOCUMENT.

THE NEXT STEP IS TO CREATE A BINARY IMAGE DOCUMENT FILE THAT IS A LISTING OF ALL MEMORY LOCATIONS AND CORRESPONDING OP CODES IN THE TARGET MEMORY DEVICE. ONE OF THESE FILES FOR EACH MEMORY DEVICE WILL BE RELEASED AS A HARD-COPY DOCUMENT OF THE CODE IN THE CHIP.

THE NEXT LINES CONTAIN DOCUMENTATION REFERENCES AT THE PROGRAMMED MEMORY DEVICE LEVEL. FOR MOST FUTURE PROJECTS, THE LOAD FILE CREATED BY THE LINKING LOADER WILL BE LARGER THAN CAN BE PROGRAMMED INTO A SINGLE CHIP. ALSO, THE LOAD FILE FOR 16-BIT PROCESSORS WILL, MOST LIKELY, HAVE TO BE LOADED INTO BYTE-ORGANIZED MEMORY. TO TAKE CARE OF THESE FACTORS, MORE SUPPORT SOFTWARE IS AVAILABLE TO SEPARATE THE SINGLE LOAD FILE INTO ODD AND EVEN BYTES, PARTITION THE LOADABLE OBJECT CODE INTO EACH PHYSICAL MEMORY DEVICE, AND FORMAT AN ASCII-HEX FILE FOR EACH MEMORY DEVICE SO ITS CORRESPONDING CODE CAN BE DOWNLOADED TO A PROM PROGRAMMER OR INTEGRATION UNIT.

8888888888888888888888888888888888888888888888888888888888888888888888
COMBINE THE FOLLOWING THREE PARAGRAPHS
8888888888888888888888888888888888888888888888888888888888888888888888
THE CODE IN EACH MEMORY DEVICE WILL BE DOCUMENTED BY A BINARY IMAGE FILE. THIS DOCUMENT IS A LISTING OF THE MEMORY LOCATIONS AND OP CODE APPLICABLE TO EACH PROGRAMMED DEVICE. EACH BINARY IMAGE FILE HAS A 722-XXXX-RN DOCUMENT NUMBER AND A DOCUMENT FILE NAME ON VAX. THIS DOCUMENT IS A PRINTOUT OF THE CODE IN THE DEVICE. ANOTHER SUPPORT SOFTWARE PACKAGE USES THIS DOCUMENTED CODE TO CREATE THE ACTUAL ASCII-HEX FILE THAT WILL BE USED TO PROGRAM EACH DEVICE. THE ASCII-HEX FILE NAME WILL BE THE NINE DIGIT DOCUMENT NUMBER FOR THE BINARY IMAGE FILE WITH A FILE EXTENSION THAT CORRESPONDS TO THE DESIRED HEX FORMAT.

THE BINARY IMAGE FILE DOCUMENT NUMBER (722-XXXX-RN) IS ASSIGNED AT

THIS TIME. BOTH THE DOCUMENT AND ASCII-HEX FILES ARE GIVEN NAMES CORRESPONDING TO TO THE SEVEN-DIGIT BINARY IMAGE FILE DOCUMENT NUMBER. THE FILE EXTENTION ON THE ASCII-HEX FILE IS SPECIFIC TO DESIRED HEX FORMAT (.HEX FOR INTEL, .LX FOR MOTOROLA, .TEK FOR TEKHEX). THE CORRESPONDING PROGRAMMED I.C. PART NUMBER IS REQUIRED SO THAT THE MASTER CHIPS CAN BE CORRECTLY LABELED.

SINCE THE LOAD FILE IS ALSO AVAILABLE IN ASCII-HEX FORMAT, IT MAY CAUSE SOME CONFUSION AS TO WHY THE ADDITIONAL FILES ARE CREATED. THE BINARY IMAGE FILE IS USED AS A HARD-COPY DOCUMENTATION OF THE CODE IN EACH MEMORY DEVICE. AS AN ADDITIONAL CHECK OF THE ACCURACY OF THIS DOCUMENT, THE BINARY IMAGE FILE IS ACTUALLY USED TO RE-CREATE A DOWNLOADABLE ASCII-HEX FILE. IT IS THIS FILE THAT IS USED TO PROGRAM THE MASTER DEVICES THAT WILL BE USED IN PRODUCTION.
88888888888888888888888888888888888888888888888888888888888888888888888888

THE FORM REQUIRES THE LISTING OF ALL TARGET SOURCE CODE FILE NAMES AND, IF APPLICABLE, THEIR CORRESPONDING RELOCATABLE OBJECT FILE NAMES. THE SOURCE FILE NAMES MUST AGREE WITH THE FILE NAMES CALLED OUT IN THE COMMAND FILE USED TO CREATE THE LOAD FILE.

THE SUPPORT SOFTWARE USED BY THE SOFTWARE DEVELOPER TO CREATE THE FINISHED VERSION OF THE LOAD FILE MUST ALSO BE DOCUMENTED. EXACT FILE NAMES AND DATES MUST BE GIVEN AND MUST AGREE WITH THE SUPPORT SOFTWARE COMMAND FILE NAMES. THE FILE NAMES THAT APPLY FOR EACH CATAGORY OF SUPPORT SOFTWARE CAN BE OBTAINED FROM SUPPORT SOFTWARE PERSONNEL. THOSE ITEMS OF SUPPORT SOFTWARE NOT USED SHOULD BE MARKED N/U.

THE FORM REQUIRES THE FOLLOWING INFORMATION FOR UPWARD REFERENCE PURPOSES:

1. THE LRU SOFTWARE DOCUMENT NUMBER. 715-XXXX-RN
2. THE PROGRAMMER'S REFERENCE LITERATURE DOCUMENT NUMBER 718-XXXX-RN
3. THE SOURCE LISTING DOCUMENT NUMBER 719-XXXX-RN
4. THE DESIGN DESCRIPTION DOCUMENT NUMBER 721-XXXX-RN
5. THE TOP-LEVEL SYSTEM NAME(S) OF ALL SYSTEMS USING THIS SOFTWARE.
6. THE LRU (UNIT) NAME(S) OF ALL LRUS CONTAINING THIS SOFTWARE.

MULTIPLE REFERENCES TO SYSTEMS AND UNITS IS REQUIRED FOR PRODUCTS THAT HAVE A GENERAL PURPOSE PROGRAM USED IN MANY APPLICATIONS.

THE BOTTOM OF THE FORM CONTAINS THE DISTRIBUTION LIST FOR THE RELEASE FORM AFTER THE RELEASE IS COMPLETED. THE DATE IS THE ACTUAL DATE THAT COPIES OF THE RELEASE FORM ARE DISTRIBUTED.

777
BEFORE THE STAGE-3 RELEASE IS ACTUALLY COMPLETED AND SIGNED OFF, MASTER COPIES OF ALL PROGRAMMED CHIPS NEED TO BE CREATED AND VERIFIED. FOR THE PRESENT TIME, THE SOFTWARE LIBRARIAN AND THE SOFTWARE PROJECT ENGINEER WILL WORK TOGETHER TO CREATE THE LOAD FILE AND BINARY IMAGE FILES ON VAX AND PROGRAM FOUR (4) MASTER COPIES OF EACH UV PROM. THESE FOUR CHIP SETS WILL BE LABELED WITH THEIR CORRESPONDING 122-XXXX-RN PART NUMBERS.

THE FIRST PART OF THE FORM DIFFERS FROM PREVIOUS RELEASES IN THAT THE PROJECT ENGINEER IS REQUIRED TO SIGN THAT HE HAS VERIFIED THE LOAD FILE PRODUCED WITH THE SUPPORT SOFTWARE COMMAND FILE. THIS VERIFICATION SHOULD INCLUDE BOTH CHIP CONTENTS AND FUNCTIONAL PERFORMANCE. AFTER VERIFICATION, THE SOFTWARE CONFIGURATION CONTROL MANAGER SIGNS THE RELEASE.

THE PROJECT ENGINEER WILL TAKE ONE OF THE CHIP SETS AND USE IT FOR VERIFICATION AGAINST THE CORRESPONDING CHIP THAT HE USES IN THE TESTED ENGINEERING MODEL. AFTER DETAILED TESTING, THE SOFTWARE PROJECT ENGINEER WILL BE REQUIRED TO SIGN OFF ON THE RELEASE FORM THAT HE HAS COMPLETED THIS VERIFICATION PROCESS.

AFTER THE VERIFICATION PROCESS HAS SHOWN THAT THE CHIPS CONTAIN THE CORRECT SOFTWARE, THE THREE REMAINING MASTER COPIES ARE SUBMITTED TO COMPONENT ENGINEERING FOR SAFEKEEPING. ONE OF THESE COPIES WILL BE KEPT AS A PERMENANT RECORD AND THE OTHER TWO COPIES WILL BE ON ON HAND WHEN RECEIVING INSPECTION NEEDS THEM.

** 6.5 PROGRAMMED I.C. SPECIFICATION NUMBER REQUEST (799-0004-00).
** 6.6 RELEASE REQUEST FOR 122-XXXX-RN DRAWING (799-0005-00).
** 6.7 SOFTWARE DOCUMENT NUMBER REQUEST (799-0007-00).
** 6.8 REQUEST FOR PROPRIETARY SOFTWARE (799-0008-00).

AFTER THE STAGE-2 RELEASE IS COMPLETED BUT BEFORE THE PURCHASING BILL-OF-MATERIALS IS RELEASED, YOU SHOULD APPLY TO COMPONENT ENGINEERING FOR THE RELEASE OF A SPECIFICATION CONTROL DRAWING FOR EACH PROGRAMMED PROCESSOR OR MEMORY CHIP IN EACH NEW LRU IN THE SYSTEM. THE PRINCIPAL REASON FOR DOING THIS IS TO OBTAIN A PART NUMBER THAT CAN BE PUT ON THE BILL-OF-MATERIALS. THIS PART NUMBER AND ITS ASSOCIATED SPECIFICATION CONTROL DRAWING WILL BE USED BY PURCHASING PLANNERS TO SCHEDULE THE DELIVERY OF ELECTICAL PARTS. THIS RELEASE REQUIRES THREE STEPS:

1. APPLY TO COMPONENT ENGINEERING FOR A 122-XXXX-RN NUMBER ASSIGNMENT.
2. APPLY TO DOCUMENT CONTROL FOR A 722-XXXX-RN DOCUMENT NUMBER FOR THE BINARY IMAGE FILE THAT WILL BE PROGRAMMED INTO THE CHIP.
3. WHEN THE BINARY IMAGE FILE DOCUMENT NUMBER HAS BEEN ASSIGNED, TAKE A COPY OF THE FORM TO COMPONENT ENGINEERING AND REQUEST THAT THE 122-XXXX-RN SPECIFICATION CONTROL DRAWING THAT APPLIES TO THAT PROGRAMMED CHIP BE RELEASED INTO THE DOCUMENT CONTROL SYSTEM.

AT THIS POINT, THE PROGRAMMED I.C. SPECIFICATION CONTROL DRAWING WILL BE AVAILABLE TO THE PURCHASING PLANNERS BUT THE SOFTWARE IS NOT YET RELEASED SO THAT THE CHIPS CAN BE PROGRAMMED.

QUANTITY PROGRAMMING OF UV PROM CHIPS FOR PRODUCTION IS ACCOMPLISHED AS FOLLOWS:

1. RECEIVING INSPECTION (R.I.) RECEIVES A WORK ORDER FROM THE PURCHASING PLANNER. THE WORK ORDER REQUESTS A QUANTITY OF PARTS TO BE WORKED UP PER A 122-XXXX-RN DRAWING. THE PLANNER ALSO PROVIDES R.I. WITH THE UNPROGRAMMED 120-XXXX-XX PARTS.

2. R.I. REQUESTS TWO COPIES OF THE MASTER CHIP FROM COMPONENT ENGINEERING. ONE OF THESE WILL BE USED AS THE SOURCE FOR GANG PROGRAMMING THE PRODUCTION DEVICES. IT WILL BE THE SO-CALLED 'WORKING COPY'. THE OTHER WILL BE MAINTAINED IN A PRISTINE CONDITION SO THAT IT MAY BE USED FOR QUALITY VERIFICATION OF THE GANG PROGRAMMING PROCESS. PROGRAMMED PARTS ARE THEN STOCKED UNDER THE 122-XXXX-RN PART NUMBER THAT APPEARS ON THE B/M.

NOTICE THAT ONCE THE PROJECT ENGINEER HAS HELPED SOFTWARE CONTROL TO PRODUCE THE FOUR MASTER COPIES, ENGINEERING IS NO LONGER INVOLVED IN THE PROGRAMMING OF CHIPS. COMPONENT ENGINEERING ALSO HAS EXTRA COPIES OF THE MASTER CHIP AVAILABLE SO THAT THE GANG PROGRAMMED PARTS CAN BE VERIFIED BY A CHIP THAT WAS NOT USED TO PROGRAM THEM BUT IS OTHERWISE IDENTICAL.

[SOFTLIB.DOC]POTTEDSCM.DEF

****
EXTREMELY CONCISE OUTLINE OF THE PROGRESSION OF SOFTWARE &
SOFTWARE DOCUMENTATION FROM THE INCEPTION OF A PROJECT.
****

THIS DOCUMENT IS INTENDED TO GIVE A GENERAL IDEA OF THE STAGES INVOLVED IN SOFTWARE DEVELOPMENT AND OF THE PART NUMBERS AND STANDARD DESCRIPTIONS WHICH MUST BE ASSIGNED AT VARIOUS STAGES. FOR MORE SPECIFIC INSTRUCTIONS AS TO WHEN AND HOW TO GENERATE PART NUMBERS AND DOCUMENTATION, REFER TO THE APPROPRIATE SOFTWARE CONTROL DOCUMENTS.

STAGE 0: THE VERY BEGINNING.
AT THIS POINT THE SCOPE OF THE PROJECT IS DEFINED AND THE RELATIVE FUNCTIONS OF SOFTWARE AND HARDWARE ARE DECIDED. THREE PART NUMBERS ARE ASSIGNED TOGETHER WITH THEIR STANDARD DESCRIPTIONS:

000-XXXX-XX	SYSTEM CID
701-XXXX-XX	SYSTEM REQUIREMENTS
705-XXXX-XX	SOFTWARE REQUIREMENTS

THE SYSTEM CID IS A 'CONFIGURATION INDEX DOCUMENT' WHICH DEFINES THE STRUCTURE OF THE SYSTEM.
THE SYSTEM REQUIREMENTS DOCUMENT DEFINES THE FUNCTIONS WHICH THE SYSTEM SHALL PERFORM.
THE SOFTWARE REQUIREMENTS DOCUMENT DEFINES THAT PART OF THE SYSTEM REQUIREMENTS WHICH SHALL BE PERFORMED BY THE SOFTWARE.
THESE DOCUMENTS WILL BE GENERATED BY THE PROJECT ENGINEER(S). THE DOCUMENT NUMBER REQUEST FORM FOR THIS STAGE IS AVAILABLE FROM SOFTWARE CONTROL.

STAGE 1: THE INDIVIDUAL UNIT STAGE.
AT THIS POINT A DECISION HAS BEEN MADE AS TO HOW MANY INDIVIDUAL UNITS ARE GOING TO MAKE UP THE SYSTEM. FOR EACH INDIVIDUAL UNIT, THE FOLLOWING NUMBERS ARE ASSIGNED (LRU STANDS FOR LINE REPLACEABLE UNIT, THAT IS AN INDIVIDUAL UNIT WITHIN A SYSTEM):

707-XXXX-XX	LRU TEST DOCUMENT
715-XXXX-XX	LRU SOFTWARE FUNCTIONS
716-XXXX-XX	LRU SOFTWARE STRUCTURE DIAGRAM

THE TEST DOCUMENT DEFINES THE MANNER IN WHICH THE SOFTWARE WILL BE INTERNALLY TESTED FOR VALIDITY.
THE FUNCTIONS DOCUMENT IS A DETAILED DOCUMENT DESCRIBING HOW THE SOFTWARE IS TO OPERATE IN A WAY THAT WILL FULFILL THE REQUIREMENTS OF THE 705-DOCUMENT.
THE STRUCTURE DOCUMENT DEFINES THE GENERAL STRUCTURE OF THE UNIT SOFTWARE.

THE SOFTWARE PROJECT ENGINEER WILL GENERATE ALL THREE OF THESE DOCUMENTS. THE DOCUMENT NUMBER REQUEST FORM FOR THIS STAGE IS AVAILABLE FROM SOFTWARE CONTROL.

STAGE 2: THE INDIVIDUAL PROCESSOR STAGE
AT THIS POINT THE SOFTWARE PROJECT ENGINEER HAS DECIDED HOW MANY PROCESSORS WILL BE NECESSARY TO FULFILL THE REQUIREMENTS ESTABLISHED BY PREVIOUS DOCUMENTS. FOR EACH PROCESSOR, THE FOLLOWING NUMBERS AND STANDARD DESCRIPTIONS ARE ASSIGNED:

718-XXXX-XX	BIBLIOGRAPHY
719-XXXX-XX	SOURCE LISTING
720-XXXX-XX	LOAD FILE RELEASE
721-XXXX-XX	DESIGN DESCRIPTION

THE BIBLIOGRAPHY IS A LISTING OF THE REFERENCE LITERATURE USED BY THE SOFTWARE PROJECT ENGINEER IN HIS DEVELOPMENT OF THE SOFTWARE. A STANDARD BIBLIOGRAPHY WITH AN ASSIGNED 718- NUMBER EXISTS FOR EACH TYPE OF PROGRAMMABLE DEVICE CURRENTLY STOCKED BY KING. THE ENGINEER SHOULD REFER TO THIS DOCUMENT TO DETERMINE WHETHER HE WISHES TO USE THE STANDARD BIBLIOGRAPHY OR WHETHER HE SHOULD ASSIST SOFTWARE CONTROL IN PREPARING A SPECIAL BIBLIOGRAPHY FOR HIS APPLICATION. IF A MICROPROCESOR SYSTEM CONTAINS MORE THAN ONE TYPE OF PROGRAMMABLE DEVICE, A 718- NUMBER MUST BE REFERENCED FOR EACH TYPE.

THE SOURCE LISTING DOCUMENT WILL BE A COPY OF THE FINAL SOURCE FILE WITH A STANDARD COVER SHEET. SOFTWARE CONTROL IS RESPONSIBLE FOR GENERATING THIS DOCUMENT.

THE LOAD FILE RELEASE WILL BE A COMPLETE LISTING OF ALL THE DOCUMENTS AND ACCESSORY INFORMATION RELEVANT TO THE PROCESSOR. IT NEED NOT BE GENERATED UNTIL THE SOFTWARE IS READY FOR RELEASE.

THE DESIGN DESCRIPTION DOCUMENT IS A DETAILED DESCRIPTION OF THE FUNCTIONAL DESIGN OF THE PROCESSOR, SHOWING THE MANNER IN WHICH IT FULFILLS THE REQUIREMENTS OF THE 715- DOCUMENT. THE SOFTWARE PROJECT ENGINEER SHOULD BE WORKING ON THIS DOCUMENT AS HE DEVELOPS HIS CODE, BUT IT NEED NOT YET BE READY FOR RELEASE.

THE DOCUMENT NUMBER REQUEST FORM FOR THIS STAGE IS AVAILABLE FROM SOFTWARE CONTROL.

THE SOFTWARE PROJECT ENGINEER SHOULD ALSO BE CREATING A COMMAND FILE WHICH HE MUST USE TO GENERATE THE CODE USED TO PROGRAM THE CHIPS USED IN ENGINEERING UNITS. THE COMMAND FILE MUST, AS A MINIMUM, CALL AN ASSEMBLER, A PARTITIONER, AND A FORMATTER. A STANDARD SAMPLE COMMAND FILE IS AVAILABLE FROM SOFTWARE CONTROL.

STAGE 3: THE INDIVIDUAL PROGRAMMABLE DEVICE STAGE

AT THIS POINT THE SOFTWARE PROJECT ENGINEER HAS DECIDED HOW MANY INDIVIDUAL DEVICES HE WILL REQUIRE TO FULFILL THE REQUIREMENTS OF THE PROCESSOR. HE SHOULD BE IN THE FINAL STAGES OF CODE DEVELOPMENT. HE SHOULD KNOW WHAT UNPROGRAMMED DEVICES HE WILL BE USING TO RECEIVE HIS CODE. HE MUST HAVE MADE THIS DECISION FOR THE PURCHASING BILL OF MATERIAL TO BE RELEASED. JUST BEFORE THE BILL OF MATERIAL IS SUBMITTED FOR RELEASE, THE FOLLOWING NUMBERS AND STANDARD DESCRIPTIONS ARE ASSIGNED FOR EACH PROGRAMMABLE DEVICE:

122-XXXX-XX	PROGRAMMED DEVICE
722-XXXX-XX	BINARY IMAGE FILE

THE PROGRAMMED DEVICE NUMBER IS THE NUMBER THAT WILL BE CALLED OUT ON THE BILL OF MATERIAL. IT DESIGNATES A PROGRAMMED PART.

THE BINARY IMAGE FILE IS A PRINTED ASCII-HEX REPRESENTATION OF THE CODE USED TO PROGRAM THE DEVICE. THE SOFTWARE PROJECT ENGINEER IS NOT RESPONSIBLE FOR GENERATING THIS DOCUMENT (IT WILL BE GENERATED BY SOFTWARE CONTROL).

THE DOCUMENT NUMBER REQUEST FORM FOR THIS STAGE IS AVAILABLE FROM SOFTWARE CONTROL.

ALL NUMBERS AND STANDARD DESCRIPTIONS ARE NOW ASSIGNED. AT THE TIME OF THE FINAL MANUFACTURING RELEASE, THE LOAD FILE RELEASE ORDER WILL BE COMPLETED. AT THIS TIME, THE DESIGN DESCRIPTION DRAFT (721- NUMBER) MUST ALSO BE SUBMITTED TO SOFTWARE CONTROL.

SOFTWARE CONTROL WILL REQUEST INFORMATION FROM THE ENGINEER SO THAT SOFTWARE CONTROL CAN GENERATE THE REMAINING NECESSARY DOCUMENTS.

AFTER RELEASE TO REVISION CONTROL

PART NUMBERING SYSTEM

WHEN SOFTWARE IS REVISED, THE LAST TWO DIGITS OF ITS KING PART NUMBER WILL INCREMENT BY ONE: IF THE PREVIOUS NUMBER WAS 122-1226-04, THE NEW NUMBER WILL BE 122-1226-05, NOT 122-1226-04 REVISION X. THIS ENSURES TRACEABILITY AND RECONSTRUCTABILITY OF ALL SOFTWARE VERSIONS.

REVISING SOFTWARE

THE MECHANISM FOR REVISING SOFTWARE IS AS FOLLOWS: AN ECO IS WRITTEN TO THE BILL OF MATERIAL WHICH CALLS OUT THE PROGRAMMED DEVICE (122-) NUMBER. A LOAD FILE RELEASE ORDER IS SUBMITTED WITH THE ECO. THE LOAD FILE RELEASE ORDER IS FORWARDED BY THE ECO REPRESENTATIVE TO SOFTWARE CONTROL, WHO WILL USE THE DOCUMENT AS FOR A NEW RELEASE TO CREATE MASTER CHIPS, PROVIDE A SPECIFICATION CONTROL DRAWING, ETC. THE SOFTWARE PROJECT ENGINEER NEED FILL IN NO FORM OTHER THAN THE LOAD FILE RELEASE ORDER FOR THE REVISION OF EXISTING SOFTWARE.

MASKING CHIPS

IN THE EVENT THAT USE OF A GIVEN PROGRAMMED DEVICE IS HIGH, PURCHASING WILL INITIATE A MASKING FEASIBILITY STUDY TO DETERMINE WHETHER IT WOULD BE COST-EFFECTIVE TO ACQUIRE THE DEVICE AS A MASKED RATHER THAN AS AN ELECTRICALLY PROGRAMMABLE PART. ENGINEERING WILL BE CLOSELY INVOLVED IN THIS STUDY. IN THE EVENT THAT THE DECISION IS MADE TO MASK THE PART, A NEW 122- PART NUMBER WILL BE ASSIGNED TO THE DEVICE. HOWEVER, THE LAST TWO DIGITS OF THE NEW PART NUMBER WILL BE THE SAME AS THE LAST TWO DIGITS OF THE CURRENT ELECTRICAL PART, SINCE THE SOFTWARE IN THE MASKED PART WILL BE THE SAME AS IT WAS IN THE ELECTRICAL PART. FOR INSTANCE, IF THE ELECTRICALLY PROGRAMMED DEVICE IS 122-1226-05 (MEANING THAT THE SOFTWARE IS IN ITS 5TH VERSION) WHEN THE DECISION IS MADE TO MASK, AND THE NEW PART NUMBER ASSIGNED IS 122-1239-00, THE PART NUMBER ACTUALLY USED FOR THE MASKED DEVICE WILL BE 122-1239-05.

PROCEDURES FOR OBTAINING A MASKED ROM

WHEN THE REQUEST FOR MASK-PROGRAMMED COMPONENT FORM HAS BEEN COMPLETED BY PURCHASING, THE DESIGN ENGINEER, AND THE COMPONENT ENGINEER, AND THE DECISION HAS BEEN MADE TO PURCHASE A MASKED PART, STEPS MUST BE TAKEN TO ENSURE THAT THE PART RECEIVED FROM THE VENDOR IS THE PROPER PART ACCORDING TO SPECIFICATIONS AND CODE CONTENT. THE FOLLOWING PROCEDURE IS TO BE USED AS A GUIDELINE TO INSURE THAT THE MASKED ROM PART FROM THE VENDOR IS CORRECT.

RECEIPT OF THE REQUEST FOR MASK PROGRAMMED COMPONENT BY THE DESIGN ENGINEER WILL SERVE AS NOTICE TO WRITE AN ECO AGAINST THE BILL OF MATERIAL THAT HAS THE EPROM PART NUMBER LISTED. THE EFFECTIVITY OF THIS ECO MUST BE NO GREATER THAN "A" WITH NOTES ADDED SPECIFYING THAT THE OLD PART IS AN ELECTRICALLY PROGRAMMED PART AND SHOULD BE PULLED FROM STOCK, ERASED, AND RESTOCKED UNDER THE NUMBER OF THE PART IN ITS UNPROGRAMMED STATE WHEN THE NEW PART BECOMES AVAILABLE.

UPON RECEIPT OF THE REQUEST FOR MASK-PROGRAMMED COMPONENT SHOWING THE AUTHORIZATION TO PURCHASE THE MASKED ROM PART, COMPONENTS ENGINEERING CREATES THE NECESSARY 122 AND 120 DRAWINGS. BEFORE CREATING THE 120 DRAWING, HOWEVER, A CHECK SHOULD BE MADE TO SEE IF A 120 ALREADY EXISTS THAT WOULD FILL THE REQUIREMENTS OF THE 122 MASKED ROM. THE NEW 122 DRAWING WILL CONTAIN THE SAME 722 NUMBER THAT WAS USED ON THE EPROM 122. IN ASSIGNING THE NEW 122 FOR THE MASKED ROM, REMEMBER THAT THE REVISION LEVEL SHOULD BE THE SAME. FOR EXAMPLE IF THE OLD EPROM PART NUMBER WAS 122-XXXX-07 THEN THE NEW MASK ROM PART NUMBER WILL BE 122-YYYY-07. AT THE SAME TIME THE DRAWINGS ARE BEING CREATED ANOTHER EPROM IS TO BE PROGRAMMED FROM THE MASTER EPROM THAT IS ALREADY ON FILE IN COMPONENTS ENGINEERING. THIS NEW EPROM SHOULD ALSO BE VERIFIED AGAINST THE SOFTWARE CONTAINED IN VAX. WHEN THE 122 HAS BEEN SIGNED OFF BY THE COMPONENT ENGINEER AND THE PROJECT ENGINEER AND RELEASED INTO THE SYSTEM, THEN THE 120, (PRELIMINARY 120 IF ONE DID NOT EXIST BEFORE) THE 122, AND THE PROGRAMMED EPROM ARE SENT TO PURCHASING.

THE DELIVERY OF THIS PACKAGE TO THE VENDOR BY PURCHASING IS DONE TWO WAYS. IF TURN-AROUND TIME IS CRITICAL THE VENDOR WILL SOMETIMES PICK UP THE PACKAGE FROM PURCHASING. OTHERWISE IT IS SENT TO THE VENDOR THROUGH THE MAIL.

WHEN THE VENDOR RECEIVES THE PACKAGE (120 AND 122 DRAWINGS AND THE PROGRAMMED EPROM) HE LOADS THE PROGRAM FROM THEEPROM INTO HIS DATA SYSTEM. HE THEN ERASES THE EPROM AND RELOADS IT FROM INFORMATION NOW CONTAINED IN HI: DATA SYSTEM. THE EPROM IS THEN SENT BACK TO EITHER PURCHASING OR COMPONENTS ENGINEERING. IF IT IS SENT TO PURCHASING THEY WILL FORWARD THE EPROM TO COMPONENT ENGINEERING.

COMPONENT ENGINEERING THEN VERIFIES THAT THE INFORMATION CONTAINED IN THE EPROM IS EXACTLY THE SAME, BIT FOR BIT, AS THAT IN THE MASTER. THEY ALSO COMPARE THE CODE IN THE EPROM AGAINST THE SOFTWARE FOR THAT EPROM CONTAINED I VAX. THE COMPONENT ENGINEER THEN HAND CARRIES THE EPROM TO THE DESIGN ENGINEERING LAB WHERE IT IS INSTALLED INTO A WORKING UNIT. (IF THERE IS NO WORKING SYSTEM WITHIN THE DESIGN ENGINEER'S LAB, IT IS HIS RESPONSIBILITY TO LOCATE A WORKING UNIT TO VERIFY THAT THE EPROM IS CORRECT.) IN ANY CASE, IF THE CODE IS NOT THE SAME AS THAT WHICH IS CONTAINED IN VAX AND THE MASTER EPROM, THEN THE EPROM FROM THE VENDOR IS ERASED, REPROGRAMMED FROM THE MASTER VERIFIED AGAINST THE MASTER AND VAX , SENT TO PURCHASING, AND THEN PURCHASING SENDS THE NEWLY PROGRAMMED EPROM BACK TO THE VENDOR WITH NOTIFICATION OF THE FAILURE. IF THE EPROM VERIFIES AGAINST THE MASTER, THE SOFTWARE IN VAX, AND WORKING UNIT, THEN COMPONENTS ENGINEERING NOTIFIES PURCHASING OF SUCH AND PURCHASING WILL THEN NOTIFY THE VENDOR TO PROCEED.

WHEN THE VENDOR RECEIVES NOTIFICATION THAT THE INFORMATION CONTAINED IN THEIR DATA SYSTEM IS CORRECT THEY WILL THEN PROCEED IN ONE OF TWO DIRECTIONS. SOME VENDORS WILL PRODUCE A NUMBER OF PROTOTYPE MASK ROM COMPONENTS BEFORE THEY BUILD THE ENTIRE LOT. IN THIS CASE, THE VENDOR WILL SEND THE PROTOTYPES TO PURCHASING OR COMPONENT ENGINEERING. IF SENT TO PURCHASING THEY WILL ROUTE THE PROTOTYPES TO COMPONENT ENGINEERING. COMPONENT ENGINEERING WILL THEN DO A BIT BY BIT CHECK AGAINST THE MASTER EPROM AND WILL ALSO COMPARE THE CODE CONTAINED IN THE MASK-ROM PART AGAINST THE SOFTWARE FOR THAT PART CONTAINED IN VAX. THE COMPONENT ENGINEER WILL THEN HAND CARRY THE PROTOTYPES TO THE DESIGN ENGINEER WHERE THE PARTS WILL BE VERIFIED IN A WORKING SYSTEM. (AGAIN, IF THERE IS NO WORKING SYSTEM IN THE DESIGN ENGINEER'S LAB, IT IS HIS RESPONSIBILITY TO LOCATE A WORKING SYSTEM AND TO VERIFY THAT THE PROTOTYPES ARE CORRECT.) IF THE PROTOTYPES ARE CORRECT, COMPONENTS ENGINEERING WILL NOTIFY PURCHASING OF SUCH AND PURCHASING WILL IN TURN NOTIFY THE VENDOR TO COMPLETE THE BALANCE OF THE ORDER.

IF THE PROTOTYPES ARE NOT VERIFIED AS BEING THE SAME AS THE MASTER EPROM, OR, IF THEY DO NOT CONTAIN THE SAME CODE AS THE SOFTWARE CONTAINED IN VAX, THEN COMPONENTS ENGINEERING WILL NOTIFY PURCHASING OF SAME WHO WILL IN TURN NOTIFY THE VENDOR OF THE PROBLEM. THE VENDOR WILL THEN PRODUCE ANOTHER PROTOTYPE LOT TO BE SENT TO COMPONENTS ENGINEERING AND TO BE VERIFIED IN THE SAME MANNER. IF NECESSARY (AS REQUESTED BY THE VENDOR) COMPONENTS ENGINEERING CAN FORWARD THE ORIGINAL EPROM TO PURCHASING WHO WILL IN TURN FORWARD IT TO THE VENDOR.

THOSE VENDORS THAT DO NOT PRODUCE A PROTOTYPE LOT WILL PRODUCE THE ENTIRE LOT (MINIMUM ORDER) AND WILL SEND THE COMPLETE ORDER TO RECEIVING INSPECTION.

UPON RECEIPT OF THE MASK ROM, RECEIVING INSPECTION WILL LOOK IN ITS FILES TO SEE IF A RELEASED 122 DRAWING EXISTS. IF THERE IS A DRAWING THAT CONTAINS VENDOR APPROVAL THEN THE PARTS ARE SPOT CHECKED AGAINST VAX TO SEE THAT THEY CONTAIN THE PROPER HEX CODE AS SPECIFIED ON THE 122 PRINT. THESE PARTS ARE ALSO CHECKED AGAINST THE MASTER EPROM THAT IS ON FILE IN RECEIVING INSPECTION. THESE PARTS ARE THEN STOCKED IN THE USUAL MANNER. IF THERE IS NO VENDOR APPROVAL ON FILE, THEN RECEIVING INSPECTION NOTIFIES COMPONENTS ENGINEERING OF THE RECEIPT OF THE MASK ROM PARTS AND FORWARDS THREE MASK ROMS TO THEM.

COMPONENTS ENGINEERING THEN VERIFIES THE MASK ROMS AGAINST THE MASTER EPROM ON FILE, AGAINST THE SOFTWARE CONTAINED IN VAX, AND THEN HAND CARRIES THE PARTS TO THE DESIGN ENGINEER WHERE THEY ARE CHECKED IN A WORKING SYSTEM. (IF THERE IS NO WORKING SYSTEM IN THE LAB IT IS THE DESIGN ENGINEER'S RESPONSIBLILITY TO FIND A WORKING SYSTEM TO VERIFY THAT THE MASK ROM IS CORRECT.) IN ANY CASE, SHOULD THE MASK ROM FAIL IN VERIFICATION, THEN COMPONENTS ENGINEERING WILL NOTIFY PURCHASING THAT THE ROM DOES NOT WORK. PURCHASING WILL IN TURN NOTIFY THE VENDOR OF SAME. IF THE MASK ROMS VERIFY AS BEING GOOD THEN COMPONENTS ENGINEERING SIGNS OFF THE PRELIMINARY 120 DRAWING AND SUBMITS THE DRAWING FOR FINAL RELEASE.

UPON RECEIPT OF THE RELEASED 120 DRAWING RECEIVING INSPECTION SPOT CHECKS THE MASK ROM PARTS AND VERIFIES THEM AGAINST THE MASTER EPROM ON FILE AND ALSO AGAINST THE SOFTWARE CONTAINED IN VAX. THE PARTS ARE THEN STOCKED II THE USUAL MANNER.

FLOW CHART FOR OBTAINING MASK-PROGRAMMED PART AFTER COST ANALYSIS APPROVAL

SEND COPIES OF COMPLETED REQUEST FOR MASK-PROGRAMMED COMPONENTS TO DESIGN ENGINEER AND COMPONENT ENGINEER. (PURCHASING)

WRITE ECO CHANGING EPROM ON BILL OF MATERIAL TO THE NEW MASK ROM PART NUMBER REQUESTED FROM COMPONENT ENGINEER. EFFECTIVITY "A". (DESIGN ENGINEER)

CREATE 122 AND PRELIMINARY 120 SPECIFICATION CONTROL DRAWING. PROGRAM AN EPROM FROM MASTER EPROM. ROUTE ALL TO PURCHASING. (COMPONENT ENGINEER)

SEND 122, 120, AND EPROM TO CHOSEN VENDOR. (PURCHASING)

SEND REPROGRAMMED EPROM TO VENDOR. (PURCHASING)

LOAD CODE INTO DATA SYSTEM, ERASE EPROM, RELOAD FROM DATA SYSTEM, ROUTE BACK TO COMPONENT ENGINEERING. (CHOSEN VENDOR)

VERIFY EPROM FROM VENDOR AGAINST MASTER EPROM AND AGAINST HEX CODE IN VAX. HAND CARRY TO DESIGN ENGINEER. VERIFY IN SYSTEM. (COMPONENT ENGINEER)

VERIFIED ??

NO

REPROGRAM EPROM AND ROUTE TO PURCHASING (COMPONENT ENGINEER)

YES

NOTIFY PURCHASING THAT EPROM FROM VENDOR IS CORRECT. (COMPONENT ENGINEER)

NOTIFY VENDOR THAT EPROM IS CORRECT. (PURCHASING)

NO

PROTOTYPES ??

YES

DEVELOP PROTOTYPE MASK-ROM MICROCOMPUTERS. SEND TO COMPONENT ENGINEER. (VENDOR)

NOTIFY VENDOR THAT PROTOTYPE DOES NOT WORK. (PURCHASING)

VERIFY AGAINST MASTER EPROM AND AGAINST HEX CODE IN VAX. HAND CARRY TO DESIGN ENGINEER AND VERIFY IN WORKING SYSTEM. (COMPONENT ENGINEER)

NO

VERIFIED ??

NOTIFY PURCHASING THAT PROTOTYPE FROM VENDOR DOES NOT WORK. (COMPONENT ENGINEER)

YES

INFORM VENDOR TO COMPLETE BALANCE OF ORDER. (PURCHASING)

BALANCE OF ORDER IS SENT TO RECEIVING INSPECTION. (VENDOR)

YES

APPROVED VENDOR ??

NO

NOTIFY COMPONENT ENGINEER OF RECEIPT OF MASK ROM FROM UNAPPROVED VENDOR. SEND 3 MASK ROMS TO COMPONENT ENGINEER. (RECEIVING INSPECTION)

VERIFY MASTER ROM AGAINST MASTER EPROM AND AGAINST HEX CODE IN VAX. HAND CARRY TO DESIGN ENGINEER AND VERIFY IN WORKING SYSTEM. (COMPONENT ENGINEER)

NO

VERIFIED ??

YES

SIGN OFF APPROVED VENDOR SECTION OF PRELIMINARY 120 DRAWING AND SUBMIT FOR FINAL RELEASE. (COMPONENT ENGINEER)

UPON RECEIPT OF RELEASED 120 DRAWING, VERIFY AGAINST MASTER EPROM AND SOFTWARE IN VAX. STOCK IN USUAL MANNER. (RECEIVING INSPECTION)

SPOT CHECK FOR VERIFICATION AGAINST MASTER EPROM AND SOFTWARE IN VAX. STOCK IN USUAL MANNER. (RECEIVING INSPECTION)

NOTIFY VENDOR THAT MASK ROM DOES NOT WORK. (PURCHASING)

NOTIFY PURCHASING THAT MASK ROM DOES NOT WORK. (COMPONENT ENGINEER)

VAX SYSTEM PROGRAMMER PROCEDURES

**NOTE: INITIAL TEXT FOR THIS IS BEING WRITTEN BY GARTH BURNS STARTING 20 JUNE 1983. (SFR)

SOFTWARE QUALITY ASSURANCE PROCEDURES

***NOTE: THIS WAS COPIED FROM CH14.LIS ON 7 JULY 1983 BY STEVE RUSSELL. IT NEEDS TO BE REVISED AND THE PROCUEDURES OUTLINED.

8888888888888888888888888888888888888888888888888888888888888888888888888888

THE SOFTWARE QUALITY ASSURANCE ORGANIZATION MUST BE INDEPENDENT OF EITHER MANUFACTURING OR ENGINEERING.

SOFTWARE QUALITY DOES NOT DEGRADE WITH USE, IT IMPROVES AS BUGS FOUND AND CORRECTED.

IN HARDWARE, A LOT OF QA IS DEVOTED TO MAKING SURE THAT THE DESIGN IS BEING MAINTAINED. IN SOFTWARE THIS IS NOT NEEDED. THE CHIPS CAN BE PERFECTLY CLONED. IN SOFTWARE IT IS THE PROBLEM OF SEEING THAT THE DESIGN HAS A MINIMUM OF BUGS, KNOWN CONFIGURATION, AND REVISION CONTROL.

NO "STANDARD" COMPONENTS.
FUNCTIONS OF QA:

1. SEE THAT DESIGN AND CODING STANDARDS ARE BEING FOLLOWED.
2. EVALUATE TEST PLAN TO SEE IF IT MEETS CRITICALITY CATEGORY.
3. MONITOR ACCEPTANCE TESTS.
3. VERIFY THAT APPROVED CONFIGURATION MANAGEMENT AND RELEASE PROCEDURES ARE BEING FOLLOWED.

SECTION 2.0 INTRODUCTION

IT HAS BEEN ESTABLISHED HISTORICALLY, THROUGH APPLICATION AND CONTROLLED STUDIES, THAT THE USE OF GOOD CONFIGURATION CONTROL PRACTICES CAN RESULT IN INCREASED QUALITY OF PRODUCT SOFTWARE. CONFIGURATION CONTROL SHOULD BE ESTABLISHED IN THREE AREAS. THE FIRST IS THE SOFTWARE DESIGN AND DEVELOPMENT ENVIRONMENT IN ENGINEERING. THE SECOND IS THE RELEASE AND MAINTANENCE OF PRODUCT SOFTWARE IN THE PRODUCTION ENVIRONMENT. THE THIRD IS THE SOFTWARE SUPPORT ENVIRONMENT. THIS CHAPTER WILL PRESENT THE CONFIGURATION MANAGEMENT PLANS FOR THESE THREE AREAS.

SECTION 2.1 PRODUCT SOFTWARE IN DEVELOPMENT

CONFIGURATION CONTROL DURING DEVELOPMENT IS INTENDED TO BE USED BY THE DEVELOPMENT GROUP TO HELP MANAGE THE ORDERLY PROGRESSION OF THE SOFTWARE FROM THE FIRST FEW MODULES TO THE FINAL INTEGRATION OF THE COMPLETED PACKAGE. INITIALLY, THE RESPONSIBILITY TO IMPLEMENT AND MONITOR THIS CONTROL WILL BE ENTIRELY WITH THE DEVELOPMENT GROUP. AS THIS TECHNIQUE MATURES AND BECOMES ESTABLISHED PRACTICE, IT IS EXPECTED THAT A MONITOR AND AUDIT FUNCTION PERFORMED BY AN INDEPENDENT QA ORGANIZATION WILL BE INITIATED.

CONFIGURATION CONTROL IN THE DEVELOPMENT ENVIRONMENT IS BASICALLY A FORMALIZED PROCEDURE FOR MAINTAINING A PROTECTED LIBRARY OF PRESENT AND PAST VERSIONS OF ALL SOURCE FILES, OBJECT FILES, LOAD FILES AND PROJECT-SPECIFIC CUSTOM SOFTWARE. A PROJECT LIBRARIAN IS ASSIGNED TO MAINTAIN THE DIRECTORY STRUCTURE AND FILES. ONLY THE LIBRARIAN HAS FULL PRIVILIGE (SP?) TO THE MAIN DIRECTORY. THE REST OF THE DESIGN TEAM MAY HAVE READ AND EXECUTE ACCESS TO THE PROTECTED LIBRARY. THE PROJECT MANAGER CAN OBTAIN INFORMATION ON THE PROJECT STATUS BY EXAMINING THE PROJECT SOFTWARE LIBRARY. THE LIBRARIAN SHOULD BE A MEMBER OF THE SOFTWARE DEVELOPMENT TEAM AND HAVE A WORKING KNOWLEDGE OF THE FUNCTION OF ALL THE MODULES AND THE PROJECT ARCHITECTURE.

SECTION 2.2 PRODUCT SOFTWARE IN PRODUCTION

AFTER THE PRODUCT SOFTWARE AND BEEN DEVELOPED AND TESTED BY THE DESIGN TEAM IT IS READY TO BE RELEASED INTO CONFIGURATION CONTROL. WHEN THIS OCCURS, THE SOFTWARE IS SUBMITTED TO A CONFIGURATION CONTROL LIBRARIAN. THE SOFTWARE CONFIGURATION CONTROL ORGANIZATION IS INDEPENDENT FROM THE DESIGN ORGANIZATIONS AND HAS THE RESPONSIBILITY OF ARCHIVING THE PRODUCT SOFTWARE AND CONTROLLING ANY FUTURE MODIFICATIONS.

SECTION 2.3 SUPPORT SOFTWARE

THE SUPPORT SOFTWARE THAT IS HOSTED ON VAX INCLUDES THE VARIOUS COMPILERS, ASSEMBLERS, LINKERS, LOCATORS, FORMATTERS, PARTITIONERS, AND RUN TIME LIBRARIES. SINCE THIS SOFTWARE IS A NECESSARY LINK TO PRODUCING THE PRODUCT MACHINE CODE, ITS CONFIGURATION MUST BE CAREFULLY MANAGED TO INSURE THAT KNOWN SUPPORT SOFTWARE PACKAGES ARE USED WITH EACH SOFTWARE RELEASE AND THAT THESE SPECIFIC VERSIONS CAN BE RETREVED FROM THE ARCHIVE AT SOME FUTURE DATE TO BE USED IN THE MAINTANENCE OF PRODUCTION SOFTWARE.

SILVER::DRA1:[SOFTLIB.DOC]APENDX12.LIS

NAME: STEVE RUSSELL
DATE: 15 MARCH 1983
REV:

GUIDELINES FOR DESIGN SOFTWARE CONFIGURATION CONTROL

AFTER MOST OF THE SOFTWARE MODULES HAVE BEEN WRITTEN AND DEBUGGED, THE PROJECT MANAGER WILL WANT TO PUT THE DESIGN SOFTWARE UNDER LOCAL CONFIGURATION CONTROL. ONE OF THE DESIGN TEAM IS DESIGNATED AS THE PROJECT LIBRARIAN AND HAS ACCESS TO A PROTECTED ACCOUNT WHERE THE VARIOUS VERSIONS OF THE SOURCE, OBJECT AND LOAD FILES ARE STORED.

THE PURPOSE OF HAVING A PROJECT SOFTWARE LIBRARY IS TO BE ABLE TO PROTECT THE SOURCE FILES THAT EXIST AT VARIOUS STAGES FROM UNCONTROLLED CHANGES AND TO KEEP A HISTORY OF CHANGES TO THE SOURCE AND TO KEEP PAST VERSIONS OF THE LOAD FILES AND THEIR STATUS.

TO START, AN ACCOUNT ON VAX IS CHOSEN TO HOST THE PROJECT SOFTWARE LIBRARY. IT MUST EXCLUDE ACCESS BY ALL PROJECT PERSONELL EXCEPT THE PROJECT SOFTWARE LIBRARIAN AND PROJECT MANAGER.

TO START, AN ACCOUNT ON VAX IS OBTAINED FROM VAX SYSTEM PERSONELL THAT USES THE LAST NAME OF THE PROJECT SOFTWARE LIBRARIAN OR A SUITABLE SUBSTITUTE. UNDER THE MAIN DIRECTORY ASSOCIATED WITH THE ACCOUNT NAME, THE LIBRARIAN SHOULD CREATE A SUBDIRECTORY USING THE PROJECT NAME FOR EACH PROJECT HE WILL MANAGE. FOR EXAMPLE, IF A LIBRARIAN WITH AN ACCOUNT NAMED BUNCH IS MANAGING PROJECTS KC-190 AND KNC-778 THE TWO SUBDIRECTORIES WOULD BE:

[BUNCH.KC190] AND [BUNCH.KNC667]

NEXT, SUBDIRECTORIES UNDER EACH PROJECT NAME SHOULD BE CREATED FOR EACH PROCESSOR SYSTEM IN THE PROJECT. FOR EXAMPLE, THE KC-190 MAY HAVE THREE PROCESSOR SYSTEMS IN IT. THE SUBDIRECTORIES MAY LOOK LIKE:

[BUNCH.KC190.LOGIC]	LOGIC PROCESSOR
[BUNCH.KC190.PITCH]	PITCH PROCESSOR
[BUNCH.KC190.ROLL]	ROLL PROCESSOR

SUBDIRECTORIES UNDER EACH PROCESSOR SYSTEM ARE NEEDED FOR THE LOAD FILE AND ITS ASSOCIATED COMMAND PROCEDURE AND FOR HOLDING TEST VERSIONS OF BOTH SOURCE CODE AND LOAD FILES. FOR EXAMPLE, THE LOGIC PROCESSOR WOULD HAVE SUBDIRECTORIES:

[BUNCH.KC190.LOGIC.LOA]
[BUNCH.KC190.LOGIC.TEST]

ADDITIONAL SUBDIRECTORIES ARE NEEDED UNDER THE MAIN PROJECT DIRECTORY TO ACCOMODATE ARCHIVE INFORMATION, SPECIAL SUPPORT SOFTWARE THAT IS SPECIFIC TO THE PROJECT, AND SIMULATION SOFTWARE. EXAMPLES FOR THE KC-190 ARE:

[BUNCH.KC190.ARCHIVE]
[BUNCH.KC190.SUPPORT]
[BUNCH.KC190.SIM]

TABLE

VERSION	DESCRIPTION
N-1	THE TESTED AND DOCUMENTED COLLECTION OF SOURCE FILES, RELOCATABLE OBJECT FILES, AND RESULTING LOAD FILE THAT WERE MOST RECENTLY ARCHIVED. THESE FILES ARE LOCATED IN THE ARCHIVE SUBDIRECTORY UNDER THE PROCESSOR SUBDIRECTORY.
N	THE COLLECTION OF SOURCE FILES, RELOCATABLE OBJECT FILES, AND RESULTING LOAD FILE THAT FORM THE CURRENT PROJECT BASELINE FOR THE PROCESSOR SOFTWARE. THIS VERSION MAY NOT BE COMPLETELY TESTED BUT TESTING IS IN PROGRESS. THESE FILES ARE LOCATED IN THE PROCESSOR SUBDIRECTORY AND THE [.LOA] SUBDIRECTORY.

N+1 SELECTED SOURCE FILES AND THEIR RELOCATABLE OBJECT FILES THAT ARE CURRENTLY UNDERGOING EXTENSIVE TESTING TO SEE IF THEY SHOULD BE INCORPORATED INTO THE CURRENT BASELINE TO

EXAMPLE:
FILE CONFIGURATION INDEX FOR KC-190

1.0 PROJECT SUBDIRECTORY: [SCM.KC190]

CONFIGURATION INDEX
1CID.LIS

2.0 PROCESSOR SUBDIRECTORIES:

2.1 LOGIC PROCESSOR: [SCM.KC190.LOGIC]

COMPILER COMMAND PROCEDURES: PASCALV00.COM
ASSEMBLER COMMAND PROCEDURE: ASSEMV00.COM

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
DELTA	DELTAV&&.PAS	DELTAV&&.MPO

2.1.1 LOAD FILE SUBDIRECTORY: [SCM.KC190.LOGIC.LOA]

COMMAND PROCEDURE	LOAD FILE	MAP FILE
LOGICV&&.COM	LOGICV&&.LOA	LOGICV&&.MAP

2.1.2 TEST SUBDIRECTORY: [SCM.KC190.LOGIC.TEST]

COMPILER COMMAND PROCEDURES: PASCAL.COM
ASSEMBLER COMMAND PROCEDURE: ASSEM.COM
LINK/LOAD COMMAND PROCEDURE: LOAD.COM

LOAD FILE
LOGICV&&.LOA

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
DELTA	DELTAV02.MPO	DELTAV02.PAS

2.2 PITCH PROCESSOR SUBDIRECTORY: [SCM.KC190.PITCH]

COMPILER COMMAND PROCEDURES: PASCALV00.COM
ASSEMBLER COMMAND PROCEDURE: ASSEMV00.COM

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
GAMMA	GAMMAV&&.PAS	GAMMAV&&.MPO

2.2.1 LOAD FILE SUBDIRECTORY: [SCM.KC190.PITCH.LOA]

COMMAND PROCEDURE	LOAD FILE	MAP FILE
PITCHV&&.COM	PITCHV&&.LOA	PITCHV&&.MAP

2.2.2 TEST SUBDIRECTORY: [SCM.KC190.PITCH.TEST]

COMPILER COMMAND PROCEDURES: PASCAL.COM
ASSEMBLER COMMAND PROCEDURE: ASSEM.COM
LINK/LOAD COMMAND PROCEDURE: LOAD.COM

LOAD FILE
PITCHV&&.LOA

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
GAMMA	GAMMAV&&.PAS	GAMMAV&&.MPO

2.3 ROLL PROCESSOR SUBDIRECTORY: [SCM.KC190.ROLL]

COMPILER COMMAND PROCEDURES: PASCALV00.COM
ASMZ8KV00.COM
ASSEMBLER COMMAND PROCEDURE: ASSEMV00.COM

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
ALPHA	ALPHAV&&.PAS	ALPHAV&&.MPO
BETA	BETAV&&.Z8K	BETAV&&.MPO

2.3.1 LOAD FILE SUBDIRECTORY: [SCM.KC190.ROLL.LOA]

COMMAND PROCEDURE	LOAD FILE	MAP FILE
ROLLV&&.COM	ROLLV&&.LOA	ROLLV&&.MAP

2.3.2 TEST SUBDIRECTORY: [SCM.KC190.ROLL.TEST]

COMPILER COMMAND PROCEDURES: PASCAL.COM
ASMZ8K.COM
ASSEMBLER COMMAND PROCEDURE: ASSEM.COM
LINK/LOAD COMMAND PROCEDURE: LOAD.COM

LOAD FILE
ROLLV&&.LOA

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
ALPHA	ALPHAV&&.PAS	ALPHAV&&.MPO

3.0 ARCHIVE SUBDIRECTORY: [SCM.KC190.ARCHIVE]

VOLUME NAME:

1. SAVE-SET NAME:
   DATE:
   FILENAME LIST:

2. SAVE-SET NAME:
   DATE:
   FILENAME LIST:

4.0 PROJECT SUPPORT-SOFTWARE SUBDIRECTORY: [SCM.KC190.SUPPORT]

5.0 PROJECT SIMULATION-SOFTWARE SUBDIRECTORY: [SCM.KC190.SIM]

APPENDIX XIII. SCM FOR SUPPORT SOFTWARE

SECTION 2.1 DEVELOPMENT SOFTWARE CONFIGURATION CONTROL

CONFIGURATION CONTROL OF PROJECT SOFTWARE

CONTROL OF THE PROJECT SOFTWARE IS NEEDED

FILE CONFIGURATION INDEX FOR KC-190

1.0 PROJECT SUBDIRECTORY: [SCM.KC190]

CONFIGURATION INDEX
1CID.LIS

2.0 PROCESSOR SUBDIRECTORIES:

2.1 LOGIC PROCESSOR: [SCM.KC190.LOGIC]

COMPILER COMMAND PROCEDURES: PASCALV00.COM
ASSEMBLER COMMAND PROCEDURE: ASSEMV00.COM

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
DELTA	DELTAV&&.PAS	DELTAV&&.MPO

2.1.1 LOAD FILE SUBDIRECTORY: [SCM.KC190.LOGIC.LOA]

COMMAND PROCEDURE	LOAD FILE	MAP FILE
LOGICV&&.COM	LOGICV&&.LOA	LOGICV&&.MAP

2.1.2 TEST SUBDIRECTORY: [SCM.KC190.LOGIC.TEST]

COMPILER COMMAND PROCEDURES: PASCAL.COM
ASSEMBLER COMMAND PROCEDURE: ASSEM.COM
LINK/LOAD COMMAND PROCEDURE: LOAD.COM

LOAD FILE
LOGICV&&.LOA

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
DELTA	DELTAV02.MPO	DELTAV02.PAS

2.2 PITCH PROCESSOR SUBDIRECTORY: [SCM.KC190.PITCH]

COMPILER COMMAND PROCEDURES: PASCALV00.COM
ASSEMBLER COMMAND PROCEDURE: ASSEMV00.COM

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
GAMMA	GAMMAV&&.PAS	GAMMAV&&.MPO

2.2.1 LOAD FILE SUBDIRECTORY: [SCM.KC190.PITCH.LOA]

COMMAND PROCEDURE	LOAD FILE	MAP FILE
PITCHV&&.COM	PITCHV&&.LOA	PITCHV&&.MAP

2.2.2 TEST SUBDIRECTORY: [SCM.KC190.PITCH.TEST]

COMPILER COMMAND PROCEDURES: PASCAL.COM
ASSEMBLER COMMAND PROCEDURE: ASSEM.COM
LINK/LOAD COMMAND PROCEDURE: LOAD.COM

LOAD FILE

PITCHV&&.LOA

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
GAMMA	GAMMAV&&.PAS	GAMMAV&&.MPO

2.3 ROLL PROCESSOR SUBDIRECTORY: [SCM.KC190.ROLL]

COMPILER COMMAND PROCEDURES: PASCALV00.COM
ASMZ8KV00.COM
ASSEMBLER COMMAND PROCEDURE: ASSEMV00.COM

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
ALPHA	ALPHAV&&.PAS	ALPHAV&&.MPO
BETA	BETAV&&.Z8K	BETAV&&.MPO

2.3.1 LOAD FILE SUBDIRECTORY: [SCM.KC190.ROLL.LOA]

COMMAND PROCEDURE	LOAD FILE	MAP FILE
ROLLV&&.COM	ROLLV&&.LOA	ROLLV&&.MAP

2.3.2 TEST SUBDIRECTORY: [SCM.KC190.ROLL.TEST]

COMPILER COMMAND PROCEDURES: PASCAL.COM
ASMZ8K.COM
ASSEMBLER COMMAND PROCEDURE: ASSEM.COM
LINK/LOAD COMMAND PROCEDURE: LOAD.COM

LOAD FILE

ROLLV&&.LOA

MODULE NAME	SOURCE FILENAME	OBJECT FILENAME
ALPHA	ALPHAV&&.PAS	ALPHAV&&.MPO

3.0 ARCHIVE SUBDIRECTORY: [SCM.KC190.ARCHIVE]

VOLUME NAME:

1. SAVE-SET NAME:
   DATE:
   FILENAME LIST:

2. SAVE-SET NAME:
   DATE:
   FILENAME LIST:

4.0 PROJECT SUPPORT-SOFTWARE SUBDIRECTORY: [SCM.KC190.SUPPORT]

5.0 PROJECT SIMULATION-SOFTWARE SUBDIRECTORY: [SCM.KC190.SIM]

APPENDIX XX. DIRECTORY STRUCTURE FOR SUPPORT SOFTWARE

THE DIRECTORY STRUCTURE FOR SUPPORT SOFTWARE HAS BEEN SET UP TO ALLOW EFFICIENT HOSTING ON VAX AND GOOD PROTECTION AND ARCHIVING PROCEDURES. THERE ARE DIRECTORIES FOR DEVELOPMENT AND MAINTANENCE OF SUPPORT SOFTWARE AND DIRECTORIES WHERE IT CAN BE TESTED AND VALIDATED. ADDITIONAL DIRECTORIES ARE USED TO ARCHIVE THE SOURCES AND TO PROVIDE WORLD ACCESS TO THE EXECUTABLE IMAGES. EXECUTABLES CAN BE TRACED BACK TO THEIR SOURCE FILES AND VERSION LEVELS ARE MAINTAINED SO THAT ANY SPECIFIC SUPPORT ENVIRONMENT THAT HAS BEEN USED IN THE PAST CAN BE RECONSTRUCTED. EXTENSIVE USE OF SUBDIRECTORY STRUCTURES IS EMPLOYED TO HELP IMPROVE CONFIGURATION MANAGEMENT AND REDUCE ERRORS.

ALL OF THE EXECUTABLE ELEMENTS AND ANCILLARY FILES NEEDED FOR THE SUPPORT SOFTWARE ENVIRONMENT ON VAX ARE LOCATED IN A PROTECTED DIRECTORY CALLED:

```
                    [ GOLD ]
MICRO$DISK :== [SILVER]::ENG:[MICROLIB]
```

ON EACH HOST VAX, THIS DIRECTORY HAS BEEN GIVEN THE LOGICAL NAME ASSIGNMENT MICRO$DISK. DIRECTORY MICRO$DISK HAS ONLY THE SUPPORT SOFTWARE THAT IS RELEASED FOR GENERAL PUBLIC USE AND HAS BEEN TESTED, VALIDATED, AND ARCHIVED. DURING THE TESTING PHASE FOR NEW VERSIONS OF SUPPORT SOFTWARE, SELECTED USERS EXECUTE THE NEW FILES OUT OF A DIRECTORY NAMED ENG:[MICROTEST] WHICH IS THE SAME ON ALL VAX HOSTS.

ACCESS TO MICRO$DISK IS LIMITED AS FOLLOWS:

```
SYSTEM,OWNER,GROUP : READ,WRITE,EXECUTE
             WORLD : READ,EXECUTE
```

THE WORLD (ALL USERS) IS GIVEN EXECUTE PRIVILEGE FOR THE EXECUTABLE IMAGES. READ PRIVILEGE IS ALSO NEEDED TO RUN THE COMMAND PROCEDURES. AT THE PRESENT TIME, THE PROTECTION ON EACH INDIVIDUAL FILE IN MICRO$DISK IS NOT CONTROLLED ALTHOUGH GENERALLY THEY ARE THE DEFAULTS USED WHEN THEY WERE CREATED. AT SOME FUTURE TIME, THESE FILE PROTECTIONS MAY ALSO BE CONTROLLED.

CURRENTLY, THE FOUR DIFFERENT FILE TYPES IN MICRO$DISK ARE:

- .EXE - EXECUTABLE IMAGE
- .COM - COMMAND PROCEDURE
- .MPO - MICROPROCESSOR RUN-TIME LIBRARY
- .Z8K - MACRO DEFINITION LIBRARY

THESE WILL NOW BE DESCRIBED IN MORE DETAIL.

THE EXECUTABLE FILES CONTAIN THE FOLLOWING SUPPORT SOFTWARE FUNCTIONS:

- ASSEMBLER COMMAND PROCESSOR
- LOADER COMMAND PROCESSOR
- SIMULATOR COMMAND PROCESSOR
- ASSEMBLERS FOR EACH UP
- LOADERS FOR EACH UP
- SIMULATORS FOR EACH UP
- ODD-EVEN BYTE SEPARATORS FOR 16-BIT UP
- LOAD FILE PARTITIONERS
- HEX FILE FORMATERS

** CURRENTLY THERE ARE NO COMPILER EXECUTABLES IN MICRO$DISK BUT THEY WILL BE ADDED IN THE NEAR FUTURE.

THE FOLLOWING TABLE SHOWS THE VARIOUS FILES EXECUTED FOR THE ASSEMBLERS, LOADERS, SIMULATORS, AND CUSTOM LOAD FILE MANAGEMENT SOFTWARE TOOLS. THE VERSION NUMBERS WILL CHANGE AS THE SUPPORT SOFTWARE IS MODIFIED.

TABLE OF EXECUTABLES IN MICRO$DISK

COMMAND PROCESSORS	*	01ASMBLR.EXE	00LOADER.EXE	SIMULATE.EXE
	*	00ASM6800.EXE	00LOA6800.EXE	SIM6800.EXE
	*	00ASM6809.EXE	00LOA6809.EXE	SIM6809.EXE
MICROPROCESSOR	*	01ASM8048.EXE	----------	SIM8048.EXE
ASSEMBLERS	*	00ASM8051.EXE	00LOA8051.EXE	SIM8051.EXE

LOADERS	*	00ASM8086.EXE	00LOA8086.EXE	SIM8086.EXE
SIMULATORS	*	00ASMZ80.EXE	00LOAZ80.EXE	SIMZ80.EXE
	*	00ASMZ8K.EXE	00LOAZ8K.EXE	SIMZ8K.EXE
CUSTOM	*	03FORMAT.EXE	(HEX FILE FORMATTER)	
TOOLS	*	02PART.EXE	(LOAD FILE PARTITIONER)	
	*	00SEPARAT.EXE	(ODD-EVEN BYTE SEPARATOR)	

THE REMAINING FILES IS MICRO$DISK WILL NOW BE GROUPED AND DESCRIBED BY FUNCTION.

FILES USED TO DOWNLOAD VIA RS-232 TO THE TEKTRONIX EMULATORS ARE:

DOWNLOAD.COM - COMMAND PROCESSOR
DOWNLOAD.EXE - EXECUTABLE IMAGE

AT THE PRESENT TIME THESE FILES ARE NOT REVISION CONTROLLED SINCE THEY ARE NOT USED IN A SOFTWARE RELEASE. AT SOME FUTURE TIME THEY MAY BE REVISION CONTROLLED TO FACILITATE BETTER MAINTAINENCE.

FILES USED TO SUPPORT THE SYSTEM KONTAC PASCAL SOFTWARE FOR THE ZILOG Z8002 MICROPROCESSOR ARE:

PASCAL.COM - COMMAND PROCESSOR
MPASCAL.EXE - EXECUTABLE IMAGE
RUNLIB.MPO - RUNTIME LIBRARY NEEDED BY THE LOADER
MACROLIB.Z8K - MACRO DEFINITION LIBRARY FOR THE ASSEMBLER

THESE ARE NOT REVISION CONTROLLED AT THE PRESENT TIME BUT THEY WILL BE IN THE NEAR FUTURE.

TWO ADDITIONAL FILES IN MICRO$DISK ARE USED TO MAKE LOGICAL NAME AND SYMBOL NAME ASSIGNMENTS AND TO INSTALL THE HIGH-USAGE IMAGES ON THE SYSTEM DISK. THEY ARE:

MICROINST.COM
AND
MICROSYM.COM.

THE PROCEDURE MICROINST.COM IS RUN AS PART OF THE SYSTEM STARTUP PROCEDURES. IT MAKES GENERIC SYSTEM-WIDE LOGICAL NAME TABLE ASSIGNMENTS FOR THE ASSEMBLER, LOADER, AND SIMULATOR FOREIGN COMMAND PROCESSORS AND FOR THE ASSEMBLERS, LOADERS, AND SIMULATORS SPECIFIC TO EACH PROCESSOR. IT ALSO RUNS SYS$SYSTEM:INSTALL TO INSTALL THE MOST RECENT AND FREQUENTLY USED IMAGES ON THE SYSTEM DISK UNDER THE DIRECTORY MICRO$DISK. IF A NEWER VERSION OF ANY OF THE SUPPORT SOFTWARE IS NEEDED BEFORE THE NEXT SYSGEN, THE VAX SYSTEM PROGRAMMER ASSIGNED TO MAINTAIN MICRO$DISK IS CONTACTED AND AN UPDATED VERSION OF MICROINST.COM IS RUN. THE PROCEDURE MICROSYM.COM IS RUN AT LOGIN TIME FOR EACH USER AND IS USED TO DEFINE SYMBOL NAMES IN THE PROCESS SYMBOL TABLE FOR ALL OF THE SUPPORT SOFTWARE ELEMENTS NEEDED BY USERS.

CURRENT CONFIGURATION MANAGEMENT AND QA POLICY DICTATES THAT NEW FILES TO BE INSTALLED IN MICRO$DISK CAN BE OBTAINED ONLY FROM THE SOFTWARE CONFIGURATION CONTROL LIBRARY SUBDIRECTORY NAMED:

SILVER::ENG:[SOFTLIB.ARCHIVE.EXE]

THIS PROTECTED AND ARCHIVED DIRECTORY AND ITS FILES ARE MAINTAINED BY THE SOFTWARE CONTROL LIBRARIAN. ALL OF THE FILES HAVE BEEN TESTED, VALIDATED, AND ARCHIVED AND CONFIGURATION CONTROLLED (IF NEEDED). ONLY THE MOST RECENT SUPPORT SOFTWARE VERSIONS ARE IN THIS DIRECTORY. IN THIS WAY, SOFTWARE CONTROL CAN DETERMINE WHAT VERSIONS OF SUPPORT SOFTWARE ARE MADE AVAILABLE FOR USE. USUALLY, THE COLLECTION OF FILES IN MICRO$DISK WILL BE DUPLICATES OF THOSE IN THE ARCHIVED DIRECTORY. OF COURSE WHEN NEWER VERSIONS ARE READY FOR RELEASE, THE VERSION LEVELS IN THE ARCHIVED DIRECTORY WILL BE HIGHER THAN THOSE IN MICRO$DISK. EACH TIME A SYSGEN IS DONE, THE FILES IN THIS SUBDIRECTORY ARE AUTOMATICALLY COPIED FROM [SOFTLIB.ARCHIVE.EXE] TO MICRO$DISK.

ALL OF THE CONFIGURATION CONTROLLED EXECUTABLES IN THE ARCHIVED DIRECTORY CAN BE RELATED BACK TO THEIR PARENT SOURCES. THESE SOURCE FILES AND ASSOCIATED COMMAND PROCEDURES THAT ARE NEEDED TO CREATE THE VALIDATED EXECUTABLES ARE LOCATED IN THE SUBDIRECTORY NAMED:

SILVER::ENG:[SOFTLIB.ARCHIVE]

AND SUBDIRECTORIES BELOW IT.

SUPPORT SOFTWARE THAT IS MICROPROCESSOR SPECIFIC IS LOCATED IN A

SUBDIRECTORY BELOW [SOFTLIB.ARCHIVE]. THE MICROPROCESSOR TYPE NUMBER IS USED AS A DIRECTORY NAME. TO ILLUSTRATE, A PARTIAL LIST OF SUBDIRECTORY NAMES IS GIVEN BELOW:

```
[SOFTLIB.ARCHIVE.6800]  - MOTOROLA 6800
[SOFTLIB.ARCHIVE.6809]  - MOTOROLA 6809
[SOFTLIB.ARCHIVE.8048]  - INTEL 8048
[SOFTLIB.ARCHIVE.8051]  - INTEL 8051
[SOFTLIB.ARCHIVE.8085]  - INTEL 8085
[SOFTLIB.ARCHIVE.I8086] - INTEL 8086/8087
[SOFTLIB.ARCHIVE.Z8002] - ZILOG Z8002
```

UNDER THE SUBDIRECTORY FOR EACH MICROPROCESSOR TYPE ARE ADDITIONAL SUBDIRECTORIES FOR SPECIFIC GROUPS OF FILES SUCH AS COMPILERS, ASSEMBLERS, LOADERS, AND SIMULATORS. FOR EXAMPLE, FOR THE ZILOG Z80, THE SUBDIRECTORY STRUCTURE IS:

```
[SOFTLIB.ARCHIVE.Z80.ASM] - ASSEMBLER SOURCE FILES
[SOFTLIB.ARCHIVE.Z80.LOA] - LOADER SOURCE FILES
[SOFTLIB.ARCHIVE.Z80.SIM] - SIMULATOR SOURCE FILES
```

A GENERAL ORGANIZATION FOR ALL FILES IN THE LOWEST SUBDIRECTORY HAS BEEN FOLLOWED FOR ALL OF THE SUPPORT SOFTWARE. THERE IS A FILE FOR EACH OF THE FOLLOWING FUNCTIONAL CATAGORIES:

- COMMAND FILE TO DOCUMENT CREATION OF THE EXECUTABLE.
- MAIN SOURCE FILES
- ALL INCLUDED FILES (FORTRAN 'INCLUDE')
- MAP FILE (WHEN DESIRED)
- EXECUTABLE FILE

TO AID IN IDENTIFICATION AND CONFIGURATION MANAGEMENT, ALL FILES HAVE A HEADER THAT CONTAINS PROGRAMMER NAME, DATE, PROGRAM TITLE, PROGRAM DESCRIPTION, NOTES, AND REVISION HISTORY.

THE COMMAND FILE DOCUMENTS AND IMPLEMENTS THE ENTIRE PROCEDURE USED TO CREATE THE EXECUTABLE IMAGE. IT REFERENCES ALL OF THE SOURCE FILES NEEDED AT COMPILE TIME (USUALLY THE MAIN PROGRAM AND 'INCLUDED' FILES) AND LINKS ALL OF THE LOCAL AND EXTERNAL OBJECTS. IT COMMANDS THE COMPILE AND LINK WITH THEIR APPROPRIATE OPTIONS AND THEN DELETES THE OBJECT FILE AND COPIES THE EXECUTABLE FILE TO SUBDIRECTORY [SOFTLIB.ARCHIVE.EXE]. THE SAME EXECUTABLE FILE IS KEPT IN TWO LOCATIONS.

TO ILLUSTRATE, WE CAN LOOK AT THE FILES IN THE LOADER SUBDIRECTORY FOR THE Z80. THEY ARE:

```
01LOAZ80.COM
01LOAZ80.FOR
01LZ80COM.FOR
01LOAZ80.EXE
```

THE COMMAND LINES FOR THE COMMAND PROCEDURE 01LOAZ80.COM ARE LISTED BELOW AND EXPLAINED:

```
$ SET VERIFY
$ FORTRAN/NOI4/NOLIST 01LOAZ80.FOR
$ LINK/CONTIG/NOMAP/NOTRACEBACK -
   01LOAZ80.OBJ, -
   [SOFTLIB.ARCHIVE.HELP]01HELP.OBJ, -
   SYS$INPUT/OPTIONS
 SYS$LIBRARY:LBRSHR/SHARE
$ DEL 01LOAZ80.OBJ;
$ COPY 01LOAZ80.EXE [SOFTLIB.ARCHIVE.EXE]
$ SET NOVERIFY
$ EXIT
```

SET VERIFY IS USED SO THAT WHEN THE PROCEDURE IS RUN, EACH LINE IS PRINTED TO THE TERMINAL SCREEN. THIS AIDS THE USER IN VERIFYING THAT THE COMMANDS EXECUTE WHAT WAS WANTED. NEXT THE MAIN FORTRAN PROGRAM IS COMPILED. NOTES IN THE HEADER TELL THE USER THAT THE FILE NAMED 01LZ80COM.FOR IS A FORTRAN COMMON AREA AND IS INCLUDED (FORTRAN 'INCLUDE') IN THE MAIN PROGRAM. NEXT, THE RESULTING OBJECT FILE IS LINKED WITH THE LOADER HELP LIBRARY AND SYSTEM LIBRARY SO THAT INTERACTIVE HELP IS AVAILABLE WHEN RUNNING THE Z80 LOADER. AFTER LINKING, THE OBJECT FILE IS DELETED SINCE IT WILL NOT BE NEEDED AGAIN AND THE RESULTING EXECUTABLE IMAGE FILE IS COPIED TO THE ARCHIVED SUBDIRECTORY NAMED [SOFTLIB.ARCHIVE.EXE]. THE VERIFY COMMAND IS RESET TO THE DEFAULT NOVERIFY.

THIS SAME BASIC SCHEME OF SUBDIRECTORY AND FILE ORGANIZATION IS FOLLOWED TO ALL OF THE 8-BIT MICROPROCESSORS THUS FAR HOSTED. THE

BASIC IDEA IS TO HAVE ALL FILES ASSOCIATED WITH THE CREATION OF AN EXECUTABLE FOR A PARTICULAR FUNCTION (I.E., ASSEMBLE, LOAD, SIMULATE, ETC.) LOCATED IN THE SAME SUBDIRECTORY. ALSO, AS NOTED BY THE EXAMPLE FOR THE Z80 LOADER, ALL FILES HAVE THE SAME VERSION NUMBER AS THE EXECUTABLE. TO INSURE THIS, THE COMMAND PROCEDURE CALLS OUT THE EXPLICIT VERSIONS OF EACH FILE. THIS IS WHY THE VERSION LEVEL IS CONTAINED IN THE FILE NAME. THE SCHEME FOR 16-BIT PROCESSORS IS MORE COMPLICATED AND HAS NOT BEEN WORKED OUT IN DETAIL. THE ORGANIZATION OF THE SOFTWARE AS IT COMES FROM THE THIRD-PARTY VENDORS VARIES CONSIDERABLY AND NO CLEAR PATTERN HAS EMERGED FOR FILE ORGANIZATION.

THE SOURCE FILES FOR THE FOREIGN COMMAND PROCESSORS ARE ALSO IN AN ARCHIVED SUBDIRECTORY CALLED:

SILVER::ENG:[SOFTLIB.ARCHIVE.COMPROCS]

UNDER THIS SUBDIRECTORY ARE ADDITIONAL SUBDIRECTORIES THAT GROUP FILES ACCORDING TO FUNCTION. THESE ARE:

[SOFTLIB.ARCHIVE.COMPROCS.ASM] - ASSEMBLER COMMAND PROCESSOR
[SOFTLIB.ARCHIVE.COMPROCS.LOA] - LOADER COMMAND PROCESSOR
[SOFTLIB.ARCHIVE.COMPROCS.SIM] - SIMULATOR COMMAND PROCESSOR

ORGANIZATION OF THE FILES IN EACH OF THESE LOWEST SUBDIRECTORIES IS SIMILIAR TO THAT USED FOR THE CORRESPONDING SUPPORT FUNCTION IN EACH MICROPROCESSOR SUBDIRECTORY.

SOURCE AND COMMAND FILES FOR CUSTOM SUPPORT SOFTWARE TOOLS SUCH AS THE SEPARATOR, PARTITIONER, AND FORMATTER ARE LOCATED IN THE ARCHIVED SUBDIRECTORY CALLED:

SILVER::ENG:[SOFTLIB.ARCHIVE.TOOLS]

AT THE PRESENT TIME, THERE IS NO SUBDIRECTORY BREAKDOWN FOR EACH INDIVIDUAL FUNCTION BUT THIS WILL PROBABLY BE DONE IN THE FUTURE. THESE CUSTOM TOOLS DO NOT NEED FOREIGN COMMAND PROCESSORS AS THEY ARE DESIGNED TO BE EXECUTED DIRECTLY. THE ORGANIZATION OF THESE FILES IS ALSO SIMILIAR TO THAT USED FOR THE MICROPROCESSOR SUBDIRECTORIES.

A CUSTOM HELP UTILITY HAS BEEN WRITTEN TO PROVIDE INTERACTIVE HELP TO LOADER AND SIMULATOR USERS. ALL FILES ASSOCIATED WITH THE HELP UTILITIES ARE LOCATED IN THE SUBDIRECTORY CALLED:

SILVER::ENG:[SOFTLIB.ARCHIVE.HELP]

THE FILE ORGANIZATION FOR THIS SUBDIRECTORY IS DIFFERENT THAN USED FOR THE OTHER SUBDIRECTORIES. IT HAS THE FOLLOWING CHARACTERISTICS:

1. ACCESS IS THROUGH SYSTEM UTILITIES SO THE MAIN HELP PROGRAM HAS BEEN WRITTEN IN VAX ASSEMBLY LANGUAGE. THE MAIN PROGRAM AND ITS RESULTING OBJECT FILE ARE:

   01HELP.MAR 01HELP.OBJ

   THIS OBJECT FILE MUST BE LINKED WITH ALL OF THE LOADERS AND SIMULATORS.

2. SINCE THE HELP INFORMATION IS UNIQUE TO EACH MICROPROCESSOR, THERE IS A SOURCE FILE (.HLP) AND ASSOCIATED LIBRARY FILE (.HLB) FOR EACH. FOR EXAMPLE, FOR THE Z80 LOADER (LZ80) AND SIMULATOR (SZ80), THE FILES ARE:

   00LZ80HLP.HLP 00LZ80HLP.HLB
   00SZ80HLP.HLP 00SZ80HLP.HLB

   THESE LIBRARIES ARE ACCESSED THROUGH THE SYSTEM UTILITIES IMPLEMENTED IN 01HELP.MAR.

CHANGE CONTROL FOR RELEASED SUPPORT SOFTWARE IS IMPLEMENTED WITH THE AID OF ADDITIONAL MAIN DIRECTORIES WHERE NEWER VERSIONS ARE CREATED AND TESTED. TO UPDATE A SUPPORT SOFTWARE FUNCTION, THE SUPPORT SOFTWARE MAINTAINER ASSIGNED TO THAT FUNCTION REQUESTS A COPY OF THE LATEST RELEASED VERSION OF THE SOURCE AND ITS SUPPORTING FILES. THESE FILES ARE COPIED TO THE DIRECTORY CALLED:

GOLD::ENG:[MICROSOFT]

IT IS IN THIS MAIN DIRECTORY AND ITS SUBDIRECTORIES THAT THE SUPPORT SOFTWARE PROGRAMMERS WORK TO MODIFY EXISTING SOFTWARE AND DESIGN NEW FUNCTIONS. WHEN THEY ARE SATISFIED THAT THE NEW OR MODIFIED DESIGN IS WORKING, THE NEW VERSION OF THE EXECUTABLE ALONG WITH ITS SUPPORTING

FILES AND INSTRUCTIONS ON HOW TO HOST IT FOR TESTING ARE TRANSFERRED TO A DIRECTORIES CALLED:

GOLD::ENG:[MICROEXCH]

THIS IS A PROTECTED DIRECTORY THAT HAS LIMITED ACCESS. FROM THIS DIRECTORY, THE NEW VERSION IS HOSTED IN A DIRECTORIES CALLED:

SILVER::ENG:[MICROTEST]
GOLD::ENG:[MICROTEST]

WHERE THEY ARE MADE AVAILABLE TO SELECTED USERS TO BE TESTED AND VALIDATED. WHEN THE SUPPORT SOFTWARE MAINTAINER AND THE USERS AND SOFTWARE CONTROL ARE SATISFIED THAT THE NEW VERSION IS CORRECT, IT IS TRANSFERED TO THE SOFTWARE CONTROL DIRECTORY TO BE ARCHIVED AND INSTALLED IN MICRO$DISK.

BIBLIOGRAPHY

BEIZER, BORIS "SOFTWARE TESTING TECHNIQUES", NEW YORK: VAN NOSTRAND REINHOLD COMPANY, 1983.

BERSOFF, EDWARD H., HENDERSON, VILAS D., AND SIEGEL, STANLEY G. "SOFTWARE CONFIGURATION MANAGEMENT", ENGLEWOOD CLIFFS, N.J.: PRENTICE-HALL, INC., 1980.

BOEHM, BARRY W. "SOFTWARE ENGINEERING ECONOMICS", ENGLEWOOD CLIFFS, N.J.: PRENTICE-HALL, INC., 1981.

EUROCAE (EUROPEAN ORGANISATION FOR CIVIL AVIATION ELECTRONICS), "SOFTWARE CONSIDERATIONS IN AIRBORNE SYSTEMS AND EQUIPMENT CERTIFICATION", DOCUMENT NO. RTCA DO-178 / EUROCAE ED-12, MAY, 1982.

DUNN, ROBERT AND RICHARD ULLMAN, "QUALITY ASSURANCE FOR COMPUTER SOFTWARE", NEW YORK: MCGRAW-HILL BOOK COMPANY, 1982.

GILLETT, WILL D. AND SEYMOUR V. POLLACK, "AN INTRODUCTION TO ENGINEERED SOFTWARE", NEW YORK: HOLT, RINEHART AND WINSTON, 1982.

JENSON, RANDALL W. AND CHARLES C. TONIES, "SOFTWARE ENGINEERING", ENGLEWOOD CLIFFS, N.J.: PRENTICE-HALL, INC., 1979.

MILLER, EDWARD AND WILLIAM E. HOWDEN, "TUTORIAL: SOFTWARE TESTING AND VALIDATION TECHNIQUES", NEW YORK: IEEE COMPUTER SOCIETY PRESS, 1981.

MYERS, GLENFORD J., "SOFTWARE RELIABILITY", NEW YORK: JOHN WILEY AND SONS, INC., 1976.

RTCA (RADIO TECHNICAL COMMISION FOR AERONAUTICS), "SOFTWARE CONSIDERATIONS IN AIRBORNE SYSTEMS AND EQUIPMENT CERTIFICATION", DOCUMENT NO. RTCA/DO-178, NOVEMBER, 1981.

REIFER, DONALD J., EDITOR, "TUTORIAL: SOFTWARE MANAGEMENT", PISCATAWAY N.J.: THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC., 1981.

SUMMERVILLE, I., "SOFTWARE ENGINEERING", LONDON: ADDISON-WESLEY PUBLISHERS LTD., 1982.

GLOSSARY OF TERMS

ASCII-HEX FILE

BINARY IMAGE FILE

CONFIGURATION CONTROL MANAGER

CONFIGURATION INDEX

CONFIGURATION ITEM

CONTAINER FILE

DOCUMENT CONTROL SYSTEM

EVEN BYTE FILE

LINE REPLACEABLE UNIT (LRU)

LOAD FILE

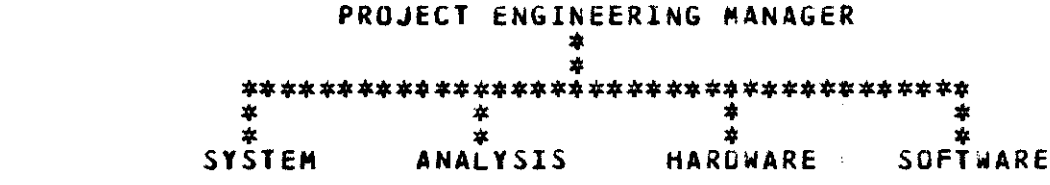
LRU

ODD BYTE FILE

PRODUCT SOFTWARE -- SOFTWARE THAT IS IMBEDDED IN THE PRODUCT AND EXECUTED BY THE TARGET PROCESSOR. PRODUCT SOFTWARE IS SOLD WITH THE PRODUCT.

PROJECT ENGINEER

PROJECT ENGINEERING MANAGER:



SOFTWARE CONFIGURATION MANAGEMENT
- SYSTEM SPECIFICATION
- DESIGN & DEVELOPMENT
- VALIDATION AND VERIFICATION
- MAINTANENCE
- DOCUMENTATION
- REVISION CONTROL
- LIFE CYCLE MANAGEMENT

SOFTWARE CONTROL LIBRARIAN

SOFTWARE ENGINEER

SOFTWARE RELEASE

SOFTWARE SUPPORT ENVIRONMENT
- TEXT EDITORS
- FILE MANAGEMENT
- COMPILERS
- ASSEMBLERS
- LINKING LOADERS
- COMMAND INTERPRETERS
- LOAD FILE PROCESSING
- EMULATORS
- INTEGRATION STATIONS
- PROM PROGRAMMERS
- ARCHIVE/BACKUP
- DOCUMENTATION
- PROGRAMMER'S REFERENCE LITERATURE

SPECIFICATION CONTROL DRAWING

SUPPORT SOFTWARE
- TEXT EDITORS
- FILE MANAGEMENT
- COMPILERS
- ASSEMBLERS
- LINKING LOADERS
- COMMAND INTERPRETERS
- LOAD FILE PROCESSING

SUPPORT SOFTWARE COMMAND PROCEDURE

DOCUMENT DEFINITIONS

SUMMARY:

000-XXXX-RN -- SYSTEM CONFIGURATION INDEX SHOWING LRUS AND INTERFACES.
PREPARED BY: PROJECT LEADER AND DESIGN TEAM.

701-XXXX-RN -- SYSTEM SPECIFICATION PROVIDING A HIGH-LEVEL DESCRIPTION OF PRODUCT INCLUDING FUNCTIONAL BLOCK DIAGRAMS, OPERATIONAL REQUIREMENTS, AND HUMAN FACTORS. DESIGN PLAN IS USED AS A SOURCE DOCUMENT.
PREPARED BY: JOINT EFFORT OF MANAGEMENT, PROJECT LEADER AND DESIGN TEAM.

702 THRU 704 -- NOT USED

705-XXXX-RN -- SOFTWARE REQUIREMENTS DOCUMENT SPECIFYING WHAT SOFTWARE IS NEEDED TO MEET SYSTEM SPECIFICATION.
PREPARED BY: PROJECT ENGINEER AND DESIGN TEAM.

706-XXXX-RN -- NOT REQUIRED

707-XXXX-RN -- SOFTWARE TEST PLAN DOCUMENT. A SEPARATE DOCUMENT IS REQUIRED FOR EACH LRU.
PREPARED BY: PROJECT ENGINEER.

708-XXXX-RN -- SOFTWARE EXECUTIVE SUMMARY
PREPARED BY: GROUP LEADER.

709 THRU 714 -- NOT USED

715-XXXX-RN -- DOCUMENT CONTAINING LRU LEVEL DESCRIPTION OF SOFTWARE. A SEPARATE DOCUMENT IS REQUIRED FOR EACH LRU.
PREPARED BY: PROJECT ENGINEER.

716-XXXX-RN -- SOFTWARE STRUCTURE DIAGRAM FOR EACH LRU.
PREPARED BY: PROJECT ENGINEER.

717 -- NOT USED

718-XXXX-RN -- DOCUMENT CONTAINING A BIBLIOGRAPHY OF PROGRAMMER'S REFERENCE LITERATURE.
PREPARED BY: SOFTWARE CONTROL.

719-XXXX-RN -- SOURCE LISTING DOCUMENT.
PREPARED BY: SOFTWARE CONTROL.

720-XXXX-RN -- LOAD FILE RELEASE DOCUMENT USED TO RECORD ALL THE INFORMATION NECESSARY TO CREATE THE EXACT VERSION OF TARGET SOFTWARE SPECIFIED. A SEPARATE DOCUMENT IS REQUIRED FOR EACH PROCESSOR.
PREPARED BY: SOFTWARE CONTROL.

721-XXXX-RN -- DESIGN DOCUMENT DESCRIBING THE DESIGN OF THE TARGET SOFTWARE. A SEPARATE DOCUMENT IS REQUIRED FOR EACH PROCESSOR.
PREPARED BY: PROJECT ENGINEER.

722-XXXX-RN -- DOCUMENT CONTAINING A FORMATTED PRINTOUT OF THE LITERAL MACHINE CODE, IN ASCII REPRESENTATION, FOR THE PROGRAMMED MEMORY DEVICE. A SEPARATE DOCUMENT IS REQUIRED FOR EACH DEVICE.
PREPARED BY: SOFTWARE CONTROL.

122-XXXX-RN -- COMPONENT SPECIFICATION FOR PROGRAMMED MEMORY DEVICE.
PREPARED BY: COMPONENT ENGINEERING.

120-XXXX-RN -- COMPONENT SPECIFICATION FOR UNPROGRAMMED ELECTRICAL DEVICE.
PREPARED BY: COMPONENT ENGINEERING.

SYSTEM CONFIGURATION INDEX
000-XXXX-RN
DO-178 REF: PARA 8.1.1.1 (SYSTEM CID)

THE PURPOSE OF THE SYSTEM CONFIGURATION INDEX DOCUMENT IS TO SHOW THE SYSTEM CONFIGURATION OF LINE REPLACEABLE UNITS (LRUS), AND THEIR INTERFACES, IN A BLOCK DIAGRAM FORMAT ON A SINGLE DRAWING.

THE SYSTEM CID IS A TOP-LEVEL LISTING FOR THE SYSTEM CONTAINING HARDWARE NAMES AND PART NUMBERS AND SOFTWARE DOCUMENT NUMBERS FOR ALL LRUS. IT IS ESSENTIALLY A BOX-LEVEL DOCUMENTATION OF ALL OF THE UNITS THAT ARE PART OF THE SYSTEM. EACH LRU IS REPRESENTED BY A BLOCK ON THE DRAWING. LRUS THAT ARE NOT NEW, NOR PRIMARY COMPONENTS OF THE NEW SYSTEM, SHOULD BE SHOWN ONLY AS INTERFACES TO THE NEW SYSTEM AND NOT AS BLOCKS ON THE DIAGRAM.

THE DOCUMENTS THAT ARE CURRENTLY USED IN KING RADIO THAT ARE SIMILIAR TO THE SYSTEM CID ARE THE PRODUCT STRUCTURE DIAGRAMS USED IN ENGINEERING AND THE CERTIFICATION DOCUMENT PACKAGES PRODUCED BY RALPH COLE'S GROUP. IT IS DESIREABLE TO HAVE A SYSTEM CID THAT SPECIFIES THE EXACT CONFIGURATION, HOWEVER, THIS IS NOT PRACTICAL WHEN LARGE NUMBERS OF SYSTEM FLAVORS ARE POSSIBLE. FOR THE PRESENT, THE RECOMMENDED APPROACH IS TO HAVE A SYSTEM CID FOR EACH 'LIKELY' SYSTEM BUT NOT TO HAVE ONE FOR EVERY POSSIBLE COMBINATION. IF THE PRACTICAL REALITY IS THAT THE NUMBER OF LIKELY SYSTEM FLAVORS IS LARGE, THE SYSTEM CID WILL HAVE TO SHOW FLAVORS AND NOT AN EXACT CONFIGURATION. EACH PROJECT ENGINEERING MANAGER (GROUP LEADER) WILL HAVE TO EVALUATE HIS PROJECT TO DETERMINE WHICH FORM OF DOCUMENT WILL BEST FIT THIS NEED.

SYSTEM SPECIFICATION
701-XXXX-RN
DO-178 REF: PARA 8.1.6 (SOURCE LISTING)

THE PURPOSE OF THE SYSTEM SPECIFICATION IS TO PROVIDE A HIGH LEVEL DOCUMENT THAT DESCRIBES THE PRODUCT AND CAN BE USED BY PROJECT PEOPLE TO DEFINE HARDWARE, SOFTWARE, AND SYSTEM REQUIREMENTS.

THE SYSTEM SPECIFICATION DOCUMENT CONTAINS BLOCK DIAGRAMS OF THE SYSTEM AND DESCRIPTIONS OF THE FUNCTIONS TO BE PERFORMED BY EACH. IT DESCRIBES THE OVERALL PERFORMANCE REQUIREMENTS AND HUMAN FACTORS CONSIDERATIONS. IT CONTAINS A DESCRIPTION OF THE DESIRED SYSTEM-LEVEL FUNCTIONS IN ENOUGH DETAIL THAT PARTITIONING OF FUNCTIONS INTO HARDWARE AND SOFTWARE IMPLEMENTATIONS CAN BE DONE LATER.

SOFTWARE REQUIREMENTS
705-XXXX-RN
DO-178 REF: PARA 8.1.2 (SOFTWARE REQUIREMENTS)

THIS DOCUMENT SHOULD CONTAIN THE FOLLOWING:

1. THE FUNCTIONAL AND OPERATIONAL REQUIREMENTS OF THE SOFTWARE, STATED IN QUANTITATIVE TERMS WITH TOLERANCES, WHERE APPLICABLE; GENERAL AND DESCRIPTIVE MATERIAL INCLUDING FUNCTIONAL BLOCK DIAGRAMS OR EQUIVALENT REPRESENTATIONS OF EACH PROCESSOR PROGRAM; GRAPHIC ILLUSTRATIONS OF FUNCTIONAL OPERATION AND THE RELATIONSHIPS BETWEEN FUNCTIONS.

2. DETAILED REQUIREMENTS FOR EACH OPERATIONAL FUNCTION OR OPERATING MODE, PLUS SPECIAL FUNCTIONS SUCH AS SEQUENCING, ERROR DETECTION AND RECOVERY, INPUT AND OUTPUT CONTROL, REAL-TIME DIAGNOSTICS, ETC.

3. PERFORMANCE, TEST, DESIGN, AND CRITICALITY REQUIREMENTS FOR EACH FUNCTION.

4. SIZING AND TIMING REQUIREMENTS.

5. HARDWARE/SOFTWARE INTERFACES.

6. BUILT-IN TEST AND/OR MONITORING FUNCTIONS.

7. PERFORMANCE (DEGRADATION AND FUNCTION) UNDER FAULT CONDITIONS.

!!! THIS DOCUMENT IS NO LONGER REQUIRED !!!

SOFTWARE MANAGEMENT PLAN
706-XXXX-RN
DO-178 REF: PARA 8.1.5

THE SOFTWARE MANAGEMENT PLAN BRIEFLY DESCRIBES THE PROJECT SCHEDULE AND OUTLINES THE REVIEWS THAT WILL BE CONDUCTED. THE PLAN ALSO DESCRIBES THE MANAGEMENT TECHNIQUES THAT WILL BE USED TO MONITOR THE DEVELOPMENT AND INSURE THAT GOOD CONFIGURATION MANAGEMENT PRACTICES WILL BE FOLLOWED. THE PROJECT MONITOR SHEET IS PART OF THIS PLAN.

LRU SOFTWARE TEST
707-XXXX-RN
DO-178 REF: PARA 8.1.11

THE LRU SOFTWARE TEST DOCUMENT DESCRIBES THE TESTS TO BE PERFORMED, THE PURPOSE OF EACH TEST, THE FUNCTIONS TO BE TESTED, AND THE SEQUENCES AND METHODS OF TESTING. IT ALSO COVERS MEANS OF VERIFICATION/VALIDATION, TEST EQUIPMENT REQUIREMENTS, TEST SOFTWARE REQUIREMENTS, AND RESULTS. THE DOCUMENT IS RELEASED IN TWO PHASES, THE FIRST BEING THE TEST PLANS AND THE SECOND BEING THE TEST RESULTS.

SOFTWARE EXECUTIVE SUMMARY
708-XXXX-RN

LRU SOFTWARE
715-XXXX-RN
DO-178 REF: PARA 8.1.1.2 (UNIT CID)

THE LRU SOFTWARE DOCUMENT CONTAINS THE FOLLOWING INFORMATION:

1. A LIST CONTAINING THE NAME DESIGNATION OF EACH PROCESSOR AND EACH SHARED-CODE MODULE IN THE LRU. EACH ITEM ON THE LIST WILL HAVE AN ASSIGNED DOCUMENT NUMBER (720-XXXX-RN).

2. A BRIEF DESCRIPTION OF THE OVERALL SOFTWARE FUNCTION OF THE LRU.

3. A BRIEF DESCRIPTION OF THE FUNCTION OF EACH PROCESSOR IN THE LRU.

4. UPWARD REFERENCE TO SYSTEM LEVEL DOCUMENTS AND REFERENCE TO THE LRU SOFTWARE STRUCTURE DIAGRAM AND TEST PLAN DOCUMENT.

IF THE SAME SOFTWARE IS USED IN SEVERAL DIFFERENT LRUS BUT IS CONFIGURED WITH HARDWARE STRAPPING OF PINS, ONLY ONE 715 DOCUMENT SHOULD BE GENERATED. THIS DOCUMENT SHOULD HAVE A SECTION EXPLAINING THE RECONFIGURATION CAPABILITY.

LRU SOFTWARE STRUCTURE DIAGRAM
716-XXXX-RN
DO-178 REF: PARA 8.1.1.2 (UNIT CID)

PROGRAMMER'S REFERENCE LITERATURE
718-XXXX-RN
DO-178 REF: PARA 8.1.4 (PROGRAMMER'S MANUAL)

THIS DOCUMENT IS A BIBLIOGRAPHY OF ALL OF THE LITERATURE USED BY THE PROGRAMMER TO GENERATE CODE FOR THE TARGET PROCESSOR. THE REFERENCES CONSISTS MAINLY OF MANFACTURER'S LITERATURE AND PROGRAMMING GUIDES AND THE SUPPORT SOFTWARE USERS GUIDES. STANDARD BIBLIOGRAPHIES FOR EACH PROCESSOR AND ASSOCIATED SUPPORT SOFTWARE ARE AVAILABLE FROM THE SUPPORT SOFTWARE GROUP. THE PROGRAMMER SHOULD GET A COPY OF THIS STANDARDIZED LIST AND USE IT AS A GUIDE TO PREPARE A BIBLIOGRAPHY OF LITERATURE THAT HE FOUND USEFUL. THIS BIBLIOGRAPHY IS THEN RELEASED INTO THE DOCUMENT CONTROL SYSTEM BY THE DESIGNER. THE ACTUAL DOCUMENT WILL BE ONLY ONE OR TWO PAGES. IT IS INTENDED TO MEET THE NEED TO HAVE A LIST OF REFERENCES THAT COULD BE USED BY NEW SOFTWARE PERSONNEL TO MODIFY RELEASED SOFTWARE.

SOURCE LISTING
719-XXXX-RN
DO-178 REF: PARA 8.1.6 (SOURCE LISTING)

THIS IS A ONE-PAGE DOCUMENT THAT REFERENCES THE FOLLOWING INFORMATION:

1. PROCESSOR NAME.
2. LOAD FILE RELEASE DOCUMENT NUMBER.
3. UNIT NAME(S).
3. SYSTEM NAME(S).

NORMALLY, THE ACTUAL SOURCE CODE IS STORED AND MAINTAINED ON MAGNETIC MEDIA AND A HARD COPY IS NOT INCLUDED WITH THE DOCUMENT. IF A PRINTOUT OF THE SOURCE CODE FILES FOR THE TARGET PROCESSOR BECOMES A REQUIRED DELIVERABLE, THE SOURCE FILES REFERENCED BY THE LOAD FILE RELEASE DOCUMENT NUMBER CAN BE RETRIEVED FROM ARCHIVE AND PRINTED. THE SOURCE LISTING DOCUMENT IS THEN USED AS A COVER SHEET FOR THIS PRINTOUT. THE LINKER/LOCATOR LISTING (WHICH PROVIDES A MAP OF THE LOAD MODULE) SHOULD ALSO BE INCLUDED.

LOAD FILE RELEASE
720-XXXX-RN
DO-178 REF: PARA 8.1.6 (SOURCE LISTING)
PARA 8.1.7 (SOURCE CODE)
PARA 8.1.8 (OBJECT CODE)

THE LOAD FILE RELEASE DOCUMENT RELEASES THE INFORMATION ON THE STAGE-3 FORM. IT CONTAINS ALL OF THE INFORMATION NEEDED TO RE-CREATED THE SAME LOAD FILE, BINARY IMAGE DOCUMENT FILES, AND ASCII-HEX FILES AT ANYTIME THROUGHOUT THE LIFETIME OF THE PRODUCT. IT ALSO REFERENCES HIGHER LEVELS OF PRODUCT DOCUMENTATION. IN SHORT, IT IS A COMPLETE PACKAGE OF DOCUMENTATION FOR ALL OF THE CODE ASSOCIATED WITH A PARTICULAR PROCESSOR (OR SHARED-CODE MODULE FOR MULTI-PROCESSOR DESIGNS). THE DOCUMENT IS PREPARED BY THE SOFTWARE CONTROL LIBRARIAN.

A LOAD FILE RELEASE DOCUMENT IS REQUIRED FOR EACH PROCESSOR AND SHARED-CODE MODULE IN THE SYSTEM. PROCESSORS THAT HAVE CODE DISTRIBUTED OVER MANY PROGRAMMED PARTS ARE STILL COVERED WITH A SINGLE DOCUMENT. THE CODE CONTAINED IN A PARTICULAR PROGRAMMED DEVICE IS SEPARATELY DOCUMENTED WITH THE BINARY IMAGE DOCUMENT.

THE LOAD FILE RELEASE DOCUMENT SHOULD CONTAIN THE FOLLOWING INFORMATION:

1. THE DATE OF RELEASE AND THE NAME OF THE SOFTWARE CONTROL LIBRARIAN.

2. THE DATE THE LOAD FILE WAS VERIFIED AND THE NAME OF THE DESIGNER THAT VERIFIED IT.

3. THE VAX FILE NAME OF THE SUPPORT SOFTWARE COMMAND PROCEDURE. THIS IS A COMMAND PROCEDURE WRITTEN BY THE SOFTWARE DESIGNER THAT CAN BE RUN BY SOFTWARE CONTROL TO PRODUCE THE LOAD FILE, ODD AND EVEN BYTE FILES (IF NEEDED), BINARY IMAGE DOCUMENT FILES, AND ASCII-HEX FILES. THIS COMMAND FILE MUST BE COMPLETE IN EVERY DETAIL AS IT BECOMES PART OF THE DOCUMENTATION PACKAGE ON HOW TO CREATE THE LOAD FILE. THE COMMAND FILE SHOULD RUN WITHOUT ANY PROMPTS OR INTERVENTION BY THE SOFTWARE CONTROL LIBRARIAN.

4. THE VAX FILE NAME FOR THE LOAD FILE. THIS IS CHOSEN BY THE SOFTWARE ENGINEER AS A DESCRIPTIVE NAME FOR THE TARGET SOFTWARE.

5. THE ARCHIVE VOLUME NAME AND DATE. THESE ARE SUPPLIED BY THE SOFTWARE CONTROL LIBRARIAN. ALL OF THE TARGET SOFTWARE WILL BE ARCHIVED ON MAGNETIC TAPE IN A CONTAINER FILE (SAVE-SET) THAT HAS A NAME CORRESPONDING TO THE LOAD FILE RELEASE DOCUMENT NUMBER, I.G. 720XXXXRN. ARCHIVING MUST BE DONE PRIOR TO FINAL DOCUMENT RELEASE.

5. A BINARY IMAGE FILE DOCUMENT NUMBER FOR EACH PROGRAMMED MEMORY DEVICE AND ITS ASSOCIATED DOCUMENT FILE NAME AND ASCII-HEX FILE NAME ON VAX.

6. FOR 16-BIT MACHINES, THE VAX FILE NAMES OF THE ODD BYTE AND EVEN BYTE FILES USES TO PROGRAM BYTE-WIDE MEMORY DEVICES.

7. ALL OF THE TARGET SOURCE CODE FILE NAMES AND THEIR CORRESPONDING RELOCATABLE OBJECT FILE NAMES. THE SOURCE FILE NAMES MUST AGREE WITH THE FILE NAMES CALLED OUT IN THE COMMAND FILE USED TO CREATE THE LOAD FILE. THESE INCLUDE HLL SOURCE CODE, ASSEMBLY SOURCE CODE, AND SPECIAL USER LIBRARIES.

8. THE SUPPORT SOFTWARE USED BY THE SOFTWARE DEVELOPER TO CREATE THE FINISHED VERSION OF THE LOAD FILE. EXACT FILE NAMES AND DATES MUST BE GIVEN AND MUST AGREE WITH THE NAMES GIVEN IN THE SUPPORT SOFTWARE COMMAND PROCEDURE. THE FILE NAMES THAT APPLY FOR EACH CATAGORY OF SUPPORT SOFTWARE CAN BE OBTAINED FROM SUPPORT SOFTWARE PERSONNEL. THOSE ITEMS OF SUPPORT SOFTWARE NOT USED SHOULD BE MARKED N/U.

9. A LIST OF HIGHER LEVEL SOFTWARE DOCUMENTS THAT WILL BE USED FOR UPWARE REFERENCE:

A. THE TOP-LEVEL SYSTEM NAME(S).
B. THE LRU (UNIT) NAME(S).
C. THE LRU SOFTWARE DOCUMENT NUMBER. 715-XXXX-RN
D. THE DESIGN DESCRIPTION DOCUMENT NUMBER 721-XXXX-RN
E. THE SOURCE LISTING DOCUMENT NUMBER 719-XXXX-RN
F. THE PROGRAMMER'S REFERENCE LITERATURE DOCUMENT NUMBER 718-XXXX-RN

10. THE COPY DISTRIBUTION FOR THE RELEASE FORM.

DESIGN DESCRIPTION
721-XXXX-RN
DO-178 REF: PARA 8.1.3 (DESIGN DESCRIPTION)

THIS DOCUMENT DESCRIBES THE DESIGN OF THE TARGET SOFTWARE AND SHOWS HOW THE DESIGN SATISFIES EACH APPLICABLE REQUIREMENT IN THE SYSTEM REQUIREMENTS DOCUMENT. THERE IS NO SET FORMAT FOR THIS DOCUMENT BECAUSE THERE ARE SEVERAL METHODS CURRENTLY IN USE THAT ARE CAPABLE OF DOING THE JOB. THE MAIN GOAL OF DOCUMENT IS TO DESCRIBE THE DESIGN IN AS MUCH DETAIL AS NECESSARY TO ALLOW SOMEONE OTHER THAN THE DESIGNER TO MODIFY AND MAINTAIN THE PROCESSOR SOFTWARE.

SPECIFICATION DO-178 RECOMMENDS THAT THE FOLLOWING INFORMATION BE INCLUDED:

1. A DESCRIPTION OF THE PROGRAM STRUCTURE (DESIGN TREES) AND PARTITIONING.

2. DATA FLOW DESCRIPTION OR DIAGRAM.

3. PROGRAM CONTROL FLOW DESCRIPTION OR DIAGRAM.

4. DESCRIPTIONS OF DATA AND CONTROL INTERFACES BETWEEN SOFTWARE PARTITIONS AND BETWEEN SOFTWARE AND HARDWARE.

5. DESCRIPTIONS OF THE ALGORITHMS.

6. TIMING SPECIFICATION.

7. MEMORY ORGANIZATION AND SIZING INFORMATION.

A DESIGN DESCRIPTION DOCUMENT IS REQUIRE FOR EACH TARGET PROCESSOR IN THE SYSTEM. THE DESIGNER MUST CREATE THIS DOCUMENT AND RELEASE IT INTO THE DOCUMENT CONTROL SYSTEM.

BINARY IMAGE FILE
722-XXXX-RN
DO-178 REF: NONE

THIS DOCUMENT IS AN OP CODE LISTING OF THE PROGRAM CONTAINED IN A PROGRAMMED MEMORY DEVICE. THE PRINTOUT IS AUTOMATICALLY GENERATED BY THE SUPPORT SOFTWARE AND CONTAINS A COMMON FORMAT OF MEMORY LOCATION AND OP CODE. THE DISTRIBUTION OF THE DOCUMENT WILL BE CONTROLLED BECAUSE IT IS COMPANY PROPRIETARY. IT WILL NOT BE AVAILABLE TO THE 'PUBLIC' FROM THE PRINT ROOM. A BINARY IMAGE FILE DOCUMENT IS REQUIRED FOR EACH PROGRAMMED MEMORY PART. THE DESIGNER MUST CREATE THIS DOCUMENT AND RELEASE IT INTO THE DOCUMENT CONTROL SYSTEM USING A STANDARD FORM FOR THE COVER SHEET AND THE PRINTOUT FROM VAX. EACH BINARY IMAGE FILE HAS A 722-XXXX-RN DOCUMENT NUMBER AND A DOCUMENT FILE NAME ON VAX. THIS DOCUMENT IS A PRINTOUT OF THE CODE IN THE DEVICE. ANOTHER SUPPORT SOFTWARE PACKAGE USES THIS DOCUMENTED CODE TO CREATE THE ACTUAL ASCII-HEX FILE THAT WILL BE USED TO PROGRAM EACH DEVICE. THE ASCII-HEX FILE NAME WILL BE THE NINE DIGIT DOCUMENT NUMBER FOR THE BINARY IMAGE FILE WITH A FILE EXTENSION THAT CORRESPONDS TO THE TARGET CPU.

STEVE RUSSELL
15 JUNE 1983

DESIGN REVIEW GUIDLINES

PDR
CDR

REVIEWS ARE NOT INVENTING SESSIONS.
REVIEWER SHOULD HAVE ~~IO~~ NO POWER TO CHANGE DESIGN BUT ONLY MAKE SUGGESTIONS.
TELL WHAT AND WHO OF REVIEWS.

****
SUGGESTED METHOD FOR DESIGNING STANDARD DESCRIPTIONS
****

IN DOCUMENTING SOFTWARE FROM THE INCEPTION OF A PROJECT, AT LEAST 11 PART NUMBERS ARE ASSIGNED ON FOUR DIFFERENT FORMS. FOR EACH PART NUMBER ASSIGNED, YOU NEED TO PROVIDE A STANDARD DESCRIPTION. IN ORDER TO FACILITATE THIS TASK FOR YOU, THIS GUIDE TO A POSSIBLE STANDARD DESCRIPTION SYSTEM FOR SOFTWARE DOCUMENTATION HAS BEEN GENERATED AND MADE AVAILABLE. YOU ARE NOT OBLIGED TO FOLLOW THIS SYSTEM; IT IS MERELY A SUGGESTED SYSTEM WHICH SOFTWARE CONTROL FEELS FULFILLS DESIRABLE DESCRIPTIVE FUNCTIONS.

THIS DOCUMENT WILL DEAL WITH POSSIBLE DESCRIPTIONS FOR PART NUMBERS IN THE ORDER IN WHICH THEY ARE ASSIGNED. IT IS SUGGESTED THAT ALL STANDARD DESCRIPTIONS CARRY A REFERENCE TO THE UNIT OR SYSTEM UNDER CONSIDERATION.

STANDARD DESCRIPTIONS MAY IN ALL CASES USE UP TO A MAXIMUM OF 18 CHARACTERS.

000-XXXX-00 SYSTEM CID

THIS DOCUMENT IS THE 'CONFIGURATION INDEX DOCUMENT'; IT DEFINES THE STRUCTURE OF THE SYSTEM TO BE DESIGNED. WE THEREFORE SUGGEST THE FOLLOWING STANDARD DESCRIPTION FORMAT:

K Y Y 9 9 9 9 S Y S T S T R — 'UNIT NAME' + 'SYSTEM STRUCTURE' (15 CHARACTERS TOTAL)

701-XXXX-00 SYSTEM REQUIREMENTS

DEFINES SYSTEM PERFORMANCE. SUGGESTED STANDARD DESCRIPTION FORMAT:

K Y Y 9 9 9 9 S Y S T R E Q — 'UNIT NAME' + 'SYSTEM REQUIREMENTS' (15 CHARACTERS TOTAL)

705-XXXX-00

DEFINES SYSTEM SOFTWARE REQUIREMNTS. SUGGESTED STANDARD DESCRIPTION FORMAT:

K Y Y 9 9 9 9 S O F T R E Q — 'UNIT NAME' + 'SOFTWARE REQUIREMENTS' (15 CHARACTERS TOTAL)

707-XXXX-00

DEFINES METHOD BY WHICH INTERNAL SELF-TESTING OF SOFTWARE WILL TAKE PLACE. SUGGESTED STANDARD DESCRIPTION FORMAT:

K Y Y 9 9 9 9 S O F T E S T — 'UNIT NAME' + 'SOFTWARE TEST' (15 CHARACTERS TOTAL)

715-XXXX-00

DEFINES SOFTWARE OPERATION AND FUNCTIONING IN DETAIL. SUGGESTED STANDARD DESCRIPTION FORMAT:

K Y Y 9 9 9 9 S O F U N C T — 'UNIT NAME' + 'SOFTWARE FUNCTIONS' (15 CHARACTERS TOTOAL)

716-XXXX-00

DEFINES THE INTERNAL STRUCTURE OF THE UNIT SOFTWARE. SUGGESTED STANDARD DESCRIPTION FORMAT:

K Y Y 9 9 9 9 S O F T S T R — 'UNIT NAME' + 'SOFTWARE STUCTURE' (15 CHARACTERS TOTAL)

718-XXXXXXX

THE BIBLIOGRAPHY FOR THE PROGRAMMABLE DEVICE BEING USED. NORMALLY YOU WILL USE THE STANDARD BIBLIOGRAPHY (AND STANDARD DESCRIPTION) AVAILABLE THROUGH SOFTWARE CONTROL. IF, HOWEVER, YOU FIND IT NECESSARY TO GENERATE A UNIQUE BIBLIOGRAPHY, WE SUGGEST THE FOLLOWING STANDARD DESCRIPTION FORMAT:

K Y Y 9 9 9 9 8 7 4 8 B I B — 'UNIT NAME' + 'DEVICE TYPE' + (16 CHARACTERS TOTAL) 'BIBLIOGRAPHY'

719-XXXX-XX

A COPY OF THE SOURCE CODE WITH A STANDARDIZED COVER SHEET. SUGGESTED STANDARD DESCRIPTION FORMAT:

9 9 9 9 A N T C P L S R C	'UNIT NAME' + 'PROCESSOR SYSTEM NAME' (15 CHARACTERS TOTAL) + 'SOURCE'

720-XXXX-XX

LISTS ALL RELATED AND ASSOCIATED DOCUMENT NUMBERS, FILE NAMES, AND STANDARD DESCRIPTIONS FOR A LOAD FILE RELEASE. SUGGESTED STANDARD DESCRIPTION FORMAT:

9 9 9 9 A N T C P L R E L	'UNIT NAME' + 'PROCESSOR SYSTEM NAME' (15 CHARACTERS TOTAL) + 'RELEASE'

721-XXXX-XX

DETAILS SOFTWARE FUNCTIONING AND DESIGN AT THE PROCESSOR LEVEL. SUGGESTED STANDARD DESCRIPTION FORMAT:

9 9 9 9 A N T C P L D S N	'UNIT NAME' + 'PROCESSOR SYSTEM NAME' (15 CHARACTERS TOTAL) + 'DESIGN'

122-XXXX-XX

THE PROGRAMMED DEVICE PART NUMBER. IN ONE-CHIP SYSTEMS, THIS STANDARD DESCRIPTION MAY WELL REFLECT THE PROCESSOR SYSTEM NAME; IN MULTIPLE-CHIP SYSTEMS, YOU WILL NEED TO COME UP WITH DIFFERENT NAMES, PREFERRABLY FUNCTION-ORIENTED, TO DISTINGUISH THEM. NOTE THE SUGGESTED USE OF 'E' OR 'M' TO DENOTE AN ELECTRICALLY PROGRAMMED VS A MASKED PART. SUGGESTED STANDARD DESCRIPTION FORMAT:

9 9 9 9 C P L V 9 9 E P G	'UNIT NAME' + 'PROCESSOR SYSTEM NAME DERIVATIVE' + 'SOFTWARE VERSION LEVEL' + 'PROGRAMMING TYPE' + 'PROGRAMMED DEVICE' (15 CHARACTERS TOTAL)

IF THE SYSTEM INCLUDED ANOTHER CHIP WHICH SERVED AS A MEMORY CHIP, ITS STANDARD DESCRIPTION MIGHT BE:

9 9 9 9 M E M V 9 9 M P G	'UNIT NAME' + 'PROCESSOR SYSTEM NAME DERIVATIVE' + 'SOFTWARE VERSION LEVEL' + 'PROGRAMMING TYPE' + 'PROGRAMMED DEVICE' (15 CHARACTERS TOTAL)

722-XXXX-XX

THE PRINTED ASCII-HEX REPRESENTATION OF THE MACHINE CODE FOR AN INDIVIDUAL CHIP. SUGGESTED STANDARD DESCRIPTION FORMAT:

9 9 9 9 C P L V 9 9 A S C	'UNIT NAME' + 'PROCESSOR SYSTEM NAME DERIVATIVE' + 'SOFTWARE VERSION LEVEL' + 'ASCII-HEX REPRESENTATION'

****
ADD VERSION LEVEL INFO TO ALL.
ADD FFS.
BYE GUYS! DEB

SOFTWARE METHODOLOGY

5.0 CONFIGURATION CONTROL

AUTHORIZATION FOR RELEASE OF DOCUMENT NUMBERS

120-XXXX-XX	COMPONENT ENGINEERING	(DALE COOPER)
122-XXXX-RN	COMPONENT ENGINEERING	(DALE COOPER)
715-XXXX-RN	ASSIGN SEQUENTIALLY	(QUICK)
718-XXXX-RN	ASSIGN SEQUENTIALLY	(QUICK)
719-XXXX-RN	ASSIGN SEQUENTIALLY	(QUICK)
720-XXXX-RN	SOFTWARE CONTROL	(STEVE RUSSELL)
721-XXXX-RN	ASSIGN SEQUENTIALLY	(QUICK)
722-XXXX-RN	SOFTWARE CONTROL	(STEVE RUSSELL)

5.1 INTRODUCTION

*** THIS NEEDS TO BE REVISED TO REFLECT HOW REVISION CONTROL IS ACUALLY DONE.**********

THIS SECTION DESCRIBES THE METHOD BY WHICH LOAD FILES ARE CREATED, DOCUMENTED, REVISED, CONTROLLED, PROTECTED AND DOWNLOADED TO PRODUCE THE MASTER CHIP SET.

PART NUMBERS (PN) OR DOCUMENT NUMBERS (DN) ARE ASSIGNED AT THE FOLLOWING LEVELS:

LRU	715-XXXX-RN
LOAD FILE	720-XXXX-RN
DESIGN DESCRIPTION	721-XXXX-RN
SOURCE LISTING	719-XXXX-RN
PROGRAMMING REFERENCES	718-XXXX-RN
BINARY IMAGE FILE	722-XXXX-RN
PROGRAMMED IC	122-XXXX-RN
UNPROGRAMMED IC	120-XXXX-XX

THE DASH NUMBER ON A SOFTWARE DN DENOTES THE REVISION LEVEL OF THE SOFTWARE AND NOT A PARTICULAR FLAVOR OF THE BASIC PRODUCT. A DIFFERENT SEVEN-DIGIT SOFTWARE DN IS ASSIGNED TO EACH FLAVOR OF THE PRODUCT.

5.2 PART NUMBER OR DOCUMENT NUMBER ASSIGNMENTS

5.2.1 LRU SOFTWARE

THERE ARE FOUR LEVELS OF DOCUMENT NUMBER ASSIGNMENTS IN THE SOFTWARE ASSOCIATED WITH EACH LRU IN A KRC PRODUCT. THE FIRST LEVEL IS THE DOCUMENT NUMBER ASSIGNED TO ALL THE SOFTWARE CONTAINED IN A PARTICULAR LRU. FOR EXAMPLE, THE SOFTWARE IN A KA-120 WITH KDN 066-1089-XX WOULD HAVE THE SOFTWARE DN 700-0001-00. THE REVISION HISTORY FOR THIS SOFTWARE WOULD BE CARRIED BY THE DASH NUMBER SO, FOR EXAMPLE, DN 700-0001-14 WOULD BE THE 14-TH REVISION OF THE SOFTWARE IN 066-1089-XX.

WHEN DIFFERENT FLAVORS FOR THE SOFTWARE IN THE KA-120 LRU 066-1089-XX ARE NEEDED, A DIFFERENT SOFTWARE DN IS ASSIGNED. FOR EXAMPLE, THE KDN 066-1089-32 MAY HAVE A SOFTWARE DN OF 700-2277-08. THE PRINT FOR THE 700-2277-08 WOULD SHOW THAT IT IS THE 8-TH REVISION OF THE LRU SOFTWARE IN 066-1089-32.

TO AID TRACEABILITY, THE PRINT WILL ALSO MAKE REFERENCE TO THE PRODUCT STRUCTURE DIAGRAM DN AND THE DOCUMENT NUMBERS OF EACH ENTITY OF PROCESSOR CODE AND SHARED CODE.

5.2.2 LOAD FILES - PROCESSOR CODE AND SHARED CODE

A DOCUMENT NUMBER IS ASSIGNED TO ALL OF THE CODE ASSOCIATED WITH EACH PROCESSOR EXCEPT, IN MULTI-PROCESSOR DESIGNS, THE SHARED CODE. SHARED CODE WILL HAVE A SEPARATE DN ASSIGNMENT. FOR EXAMPLE, IF THE LRU 066-1089-32 HAS TWO PROCESSORS AND THEY SHARE SOME CODE, THE DOCUMENT NUMBERS MAY LOOK LIKE:

PROCESSOR A:	DN 710-5436-06
PROCESSOR B:	DN 710-5438-07
SHARED CODE:	DN 710-5437-11

THIS DOCUMENT NUMBER ASSIGNMENT DESIGNATES THE LOAD FILE

ASSOCIATED WITH EACH GROUP OF CODE. A PRINT DENOTING A LOAD FILE WILL CONTAIN THE FOLLOWING INFORMATION.

A. DETAILED DESIGN DESCRIPTION.
B. FLOW CHART.
C. VAX SOURCE FILE NAMES.
D. VAX OBJECT FILE NAMES.
E. SUPPORT SOFTWARE FILE NAMES.
F. SUPPORT SOFTWARE COMMAND FILES
G. LOAD FILE NAME.
H. BINARY IMAGE DOCUMENT NUMBER AND VAX FILE NAMES.

TO AID TRACEABILITY, THE PRINT WILL ALSO MAKE REFERENCE TO THE SOFTWARE DN FOR THE LRU (IN THIS EXAMPLE 700-2277-08)

THE LEVEL OF DETAIL CONTAINED IN THIS PRINT MUST BE SUCH THAT THE EXACT LOAD MODULE CAN BE RECREATED AT ANY TIME THROUGHOUT THE LIFETIME OF THE PRODUCT.

5.2.3 BINARY IMAGE FILES

WHEN THE LOAD FILE MUST BE DISTRIBUTED ACROSS SEVERAL CHIPS, A BINARY IMAGE FILE IS CREATED FOR THE MEMORY IMAGE IN EACH MEMORY CHIP. A DOCUMENT NUMBER IS ASSIGN TO EACH BINARY IMAGE FILE ASSOCIATED WITH THE LOAD FILE. FOR EXAMPLE, IF PROCESSOR A USES THREE MEMORY CHIPS, THE DOCUMENT NUMBERS MAY LOOK LIKE:

PROCESSOR A, BIN1	DN 750-3472-04
PROCESSOR A, BIN2	DN 750-3473-02
PROCESSOR A, BIN3	DN 750-3474-11

THE PRINT FOR THESE DOCUMENT NUMBERS CONTAINS A TABLE OF THE HEXIDECIMAL REPRESENTATION ON THE MEMORY CONTENTS FOR EACH MEMORY LOCATION. THIS PRINT IS THE OFFICIAL DEFINITION FOR THE PROGRAM SEGMENT LOCATED IN THE CHIP AND IS USED TO DEFINE THE MASKED ROM PART.

5.2.4 MEMORY CHIP

EACH PROGRAMMED MEMORY CHIP WILL HAVE A PART NUMBER WHICH DESIGNATES THE PROGRAMMED PART AND IS THE PART NUMBER WHICH APPEARS ON THE BILL OF MATERIALS.

FOR UV PROM PARTS, ENGINEERING MUST DETERMINE A TARGET DATE FOR CHANGEOVER TO A MASKED ROM.

5.9 PROCEDURE FOR ARCHIVING

THE LOAD FILE AND ALL VAX FILES THAT ARE USED TO CREATE IT ARE ARCHIVED AT THE SAME TIME THAT THE LOAD FILE IS CREATED AND RELEASED. THE LOAD FILE SOFTWARE RELEASE FORM IS USED TO RECORD THE ARCHIVE VOLUME NAME AND DATE. THE BACKUP UTILITY IS USED SO THAT A BACKUP CONTAINER FILE IS CREATED WITH A NAME THAT IS THE SAME AS THE LOAD FILE DOCUMENT NUMBER.
TWO LOCATIONS SHOULD CONTAIN THE ARCHIVE SOFTWARE.
A PERIODIC SCHEDULE OF SENDING TAPES TO SECOND ARCHIVE LOCATION NEEDS TO BE DEVELOPED.

5.10 SOFTWARE DOCUMENT STRUCTURE DO-178

SOFTWARE METHODOLOGY DOCUMENT	12
SYSTEM CONFIGURATION INDEX DOCUMENT	1
SYSTEM REQUREMENTS DOCUMENT	13
SOFTWARE EXECUTIVE SUMMARY	10
SYSTEM SOFTWARE REQUIREMENTS DOCUMENT	2
SOFTWARE TEST PLANS/PROCEDURES AND RESULTS	11
SOFTWARE CONFIGURATION MANAGEMENT AND QA PLAN	5
LRU SOFTWARE DOCUMENTS	1
DESIGN DESCRIPTION DOCUMENTS	3
SOURCE LISTING DOCUMENTS	6
PROGRAMMER REFERENCE LITERATURE DOCUMENTS	4
LOAD FILE RELEASE DOCUMENTS	7,8
BINARY IMAGE FILE DOCUMENTS	7,8

SOFTWARE
SILVER::DRA1:[SOFTLIB.DOC]MEMO5.LIS

STEVE RUSSELL
1 NOV 1982

MEMO TO:

HANNEMAN
STOLPMAN
CAROCARI
WROBLEWSKI
LEHFELDT
SPECIAL PRODUCTS

DATE: 1 NOV 1982

FROM: GARY BURRELL

SUBJECT: SUMMARY SHEET OF IMBEDDED SOFTWARE PRODUCT AT KRC.

ENCLOSED IS A LIST OF KRC PRODUCTS WHICH CONTAIN SOFTWARE. WE ARE TRYING TO DEVELOP AN ACCURATE CROSSREFERENCE OF UNIT DESIGNATIONS AND THE PROGRAMMED MEMORY DEVICES THEY CONTAIN. PLEASE REVIEW THE LIST FOR COMPLETNESS AND ACCURACY AND RETURN TO ME BY XX OCTOBER 1982.

GARY BURRELL

COPIES: STEVE RUSSELL
NAME: JERRY DIGMAN [DIGMAN]SOFT2.DOC
NAME: STEVE RUSSELL [STEVE]SOFT2.DOC
DATE: 1 OCT 1982
REV: 22 NOV 1982

UNIT	PART NUMBER	DEVICE	SOFT. PROJ. ENGR./LAB
KN 53	120-2026-00	8048	DAVE CASEY/HANNEMAN
KST 75	120-3156-00**		MATT WHITING/SPD
KST 75	120-2104-00**	8748	MATT WHITING/SPD
KT 79	120-2084-00	8048	JIM LYALL/STOLPMAN
KNS 80	120-2025-05	68316	DEL BRANDLEY/CAROCARI
KNS 80	120-2025-06	68316	DEL BRANDLEY/CAROCARI
KNS 81	120-2045-01	6846	DEL BRANDLEY/CAROCARI
KNS 81	120-2046-00	6846	DEL BRANDLEY/CAROCARI
KNS 81	120-2083-00	52116	DEL BRANDLEY/CAROCARI
KNS 81	120-2082-00	52132	DEL BRANDLEY/CAROCARI
KR 87	120-2038-01	3870	TOM MCBRIDE/CAROCARI
KMC 90	----------	----	LARRY BEASON/HANNEMAN
KGR 95	SEE KMC 95		
KGR 95R	SEE KMC 95R		
KGR 95X	SEE KMC 95X		
KMC 95	120-2105-31#	8748	TOM DENNIS/WROBLEWSKI
KMC 95	122-0010-00	8755	TOM DENNIS/WROBLEWSKI
KMC 95	122-0011-00	8748	TOM DENNIS/WROBLEWSKI
KMC 95R	120-2105-31	8748	TOM DENNIS/WROBLEWSKI
KMC 95X	SEE KMC 95		
KMC 96	120-2086-30	8748	TOM DENNIS/WROBLEWSKI
KPT 100	120-2122-00	8048	MATT WHITING/SPD
KA 120	120-2072-10	8749	JOHN CULLER/HANNEMAN
KA 120	122-0023-00	8048	JOHN CULLER/HANNEMAN
KTS 143	120-2088-00	2716	TOM MCBRIDE/CAROCARI
KTS 143	120-2089-00	2716	TOM MCBRIDE/CAROCARI
KTS 143	120-2090-00	2716	TOM MCBRIDE/CAROCARI
KTS 143	120-2091-00	2716	TOM MCBRIDE/CAROCARI
KTS 158	120-2120-00**		RONNIE HIGGINS/LEHFELD
KTS 158	120-2121-00**		RONNIE HIGGINS/LEHFELD
KTS 158	122-0016-01	2716	RONNIE HIGGINS/LEHFELD
KTS 158	122-0017-00	2716	RONNIE HIGGINS/LEHFELD
KX 155/165	120-2037-01	MK3870	LARRY BEASON/HANNEMAN
KX 155/165	120-2094-00**	MK3870	LARRY BEASON/HANNEMAN
KX 155/165	120-2094-01**	MK3870	LARRY BEASON/GRILLOT
KX 155/165	120-2094-02	MK3870	LARRY BEASON/HANNEMAN
KC 190/1/2 MOD. 2	120-2072-12**	8749H	ART ERCOLANI/LEHFELDT
KC 190/1/2 MOD. 2	120-2102-11**	2732	ART ERCOLANI/LEHFELDT
KC 190/1/2 MOD. 2	120-2102-22**	2732	ART ERCOLANI/LEHFELDT
KC 190/1/2 MOD. 2	122-0000-00	8749H	ART ERCOLANI/LEHFELDT
KC 190/1/2 MOD. 2	122-0001-00	2732A	ART ERCOLANI/LEHFELDT
KC 190/1/2 MOD. 2	122-0003-00	2732A	ART ERCOLANI/LEHFELDT
KY 196/E	120-2032-01	8048	JEFF MEEDER/HANNEMAN
KY 197/A	----------	8049	LARRY BEASON/HANNEMAN
KI 229	120-2047-01	3870	TOM MCBRIDE/CAROCARI
KI 244	120-2067-00	8048	DAVE CASEY/STOLPMAN

KI	244	120-2067-01	8048	DAVE CASEY/STOLPMAN
KI	244	120-2068-00	8048	DAVE CASEY/STOLPMAN
KI	244	120-2069-00	8048	JIM LYALL/STOLPMAN
KI	244	122-0020-00	8048	JIM LYALL/STOLPMAN
KAS	297	120-2034-02**	8755	RON HIGGINS/LEHFELDT
KAS	297	120-2034-03**	8755	RON HIGGINS/LEHFELDT
KAS	297	120-2036-20	8755	RON HIGGINS/LEHFELDT
KAS	297A	120-2036-00**	8755	BOEN GO/LEHFELDT
KAS	297A	120-2036-10	8755	BOEN GO/LEHFELDT
KAS	297A	120-2055-10	8755	BOEN GO/LEHFELDT
KAS	297B	----------	----	RONNIE HIGGINS/LEHFELDT
KAS	297D	----------	----	ART ERCOLANI/LEHFELDT
KGR	356	----------	27128	CARLETON GAMET/STOLPMAN
KGR	356	----------	2764	CARLETON GAMET/STOLPMAN
KGR	356	----------	8048	CARLETON GAMET/STOLPMAN
KGR	356	----------	8048	CARLETON GAMET/STOLPMAN
KGR	356	----------	8048	CARLETON GAMET/STOLPMAN
KGR	356	----------	8049	CARLETON GAMET/STOLPMAN
KGR	356	----------	8049	CARLETON GAMET/STOLPMAN
KSG	414	----------	----	CHARLEY HETT/LEHFELDT
KDA	429	----------	8751	GEORGE YANTIS/LEHFELDT
KDA	430	----------	8031	DAVE FURLONG/LEHFELDT
KMC	440	----------	----	ART ERCOLANI/LEHFELDT
KMC	445	----------	----	N.A./LEHFELDT
KAH	450	----------	----	MARK ASPLUND/LEHFELDT
KAD	480	----------	----	INWOO YOON/LEHFELDT
KDC	481	----------	----	KAREN RETHMEYER/LEHFELI
KSG	483	----------	----	HAROLD JARRET/LEHFELDT
KPI	5538	----------	8748	GEORGE YANTIS/LEHFELDT
KFS	564/598	120-2094-00**	MK3870	LARRY BEASON/HANNEMAN
KFS	564/598	120-2094-01**	MK3870	LARRY BEASON/HANNEMAN
KFS	564/598	120-2094-02	MK3870	LARRY BEASON/HANNMEAN
KFS	564A/598A/599A	120-2123-00	8751	JIM BROCKERT/HANNEMAN
KDI	572B	----------	8748	GEORGE YANTIS/LEHFELDT
KFS	576	120-2053-00	8048	LEO MOORE/STOLPMAN
KNI	582	120-2057-00	3870	TOM MCBRIDE/CAROCARI
KNI	582	120-2065-01	3870	TOM MCBRIDE/CAROCARI
KFS	586	120-2059-00	3870	TOM MCBRIDE/CAROCARI
KFS	586A	122-0018-00	8049	STEVE MONNIG/HANNEMAN
KA	594	122-0004-00	8755	TOM DENNIS/WROBLEWSKI
KA	594	122-0005-00	8748	TOM DENNIS/WROBLEWSKI
KFS	594	----------	8751	TOM DENNIS/WROBLEWSKI
KNR	665	120-2044-00#	9218	DEL BRANDLEY/CAROCARI
KNR	665	120-2006-15#	3808	DEL BRANDLEY/CAROCARI
KNR	665	120-2006-20#	3809	DEL BRANDLEY/CAROCARI
KNR	665	120-2006-25#	3810	DEL BRANDLEY/CAROCARI
KNR	665	120-2006-30#	3810	DEL BRANDLEY/CAROCARI
KNR	665	120-2006-35#	3810	DEL BRANDLEY/CAROCARI
KNR	665	120-2007-30#	3501	DEL BRANDLEY/CAROCARI
KNR	665A	120-2044-01	AMD9218	DEL BRANDLEY/CAROCARI
KNR	667	----------	----	TOM MCBRIDE/CAROCARI
KDA	692	120-2048-00	8748	DOUG SMITH/WROBLEWSKI
KDA	692	120-2048-01	8748	DOUG SMITH/WROBLEWSKI
KDM	706A	122-0015-00	8749	JIM LYALL/STOPLMAN
KTR	707	120-2092-00	8748	RALPH BAZIL/SPD
KDF	806	120-2064-00	3870	TOM MCBRIDE/CAROCARI
KCU	951	120-2028-07	3870	CHARLEY HETT/WROBLEWSK
KAC	952-00/01	120-2066-30	8748	TOM DENNIS/WROBLEWSKI
KAC	952-02	120-2066-32	8748	TOM DENNIS/WROBLEWSKI
KCU	971	----------	6809	MICKEY O'DELL/SPD
KCU	972	----------	6809	BILL PHILLIPS/SPD
KCU	973 (CNTL)	----------	6809	TOM GIBSON/SPD
KCU	973 (SELAOR)	----------	8088	RALPH BAZIL/SPD
KAC	992	----------	8748	TALLY/WROBLEWSKI
KTR	993	SEE KMC 95		
KNR	6030	120-2029-00/02	82S181F	JERRY WATSON/HANNEMAN
KNR	6030	120-2056-00	8316	JERRY WATSON/HANNEMAN

** : NO LONGER IN PRODUCTION
# : OBSOLETE
-- : KPN NOT ASSIGNED YET/DEVICE NOT YET KNOWN

TO: DISTRIBUTION
FROM: STEVE RUSSELL
KOREY WILLNAUER
PAUL SALAMON
RE: SUPPORT SOFTWARE PERFORMANCE
DATE: 6 APRIL, 1983

WE NOW HAVE A PROCEDURE FOR REPORTING PERFORMANCE OBSERVATIONS FOR SUPPORT SOFTWARE. THESE OBSERVATIONS WOULD INCLUDE:

1) OBSERVED PROBLEMS OR ERRORS.
2) SUGGESTIONS FOR PERFORMANCE ENHANCEMENTS.
3) SUGGESTIONS FOR CHANGES IN THE PROCEDURES REQUIRED TO USE THE SOFTWARE.
4) SUGGESTED DOCUMENTATION CHANGES.

THE PROCEDURE IS INITIATED WHEN A USER FILLS OUT A PERFORMANCE REPORT (SAMPLE BLANK FORM ENCLOSED) AND SENDS IT TO SOFTWARE CONTROL. SOFTWARE CONTROL DELIVERS IT TO A SOFTWARE MAINTAINER AND THEY DISCUSS PROPOSED ACTIONS TO BE TAKEN. AFTER THE ALTERNATIVES ARE DISCUSSED, THE MAINTAINER CHOOSES THE BEST ONE AND BEGINS IMPLEMENTING THE CHANGES. TO NOTIFY THE USER COMMUNITY OF A PENDING ACTION, THE MAINTAINER WILL ENTER A NOTICE OF OBSERVATION, THE PROPOSED ACTION, AND A DATE TO BE COMPLETED INTO A FILE NAMED GOLD::[XXXXXXXXXXX]BUGLIST.LIS. THIS IS A PUBLIC FILE AND CAN BE ACCESSED BY ALL USERS. THE FILE HELPS KEEP THE USERS UP TO DATE ON SUPPORT SOFTWARE CHANGES. AFTER THE MODIFICATIONS HAVE BEEN COMPLETED, A DESCRIPTION OF THE EXACT ACTION TAKEN AND THE EFFECT ON USERS IS ENTERED INTO A FILE CALLED GOLD::[XXXXXXXXXXX]BUGFIX.LIS. THIS FILE CAN ALSO BE ACCESSED BY ALL USERS TO DETERMINE WHAT CHANGES HAVE OCCURRED. THE MAINTAINER THEN DOCUMENTS THESE CHANGES ON A PERFORMANCE RESPONSE (SAMPLE BLANK ENCLOSED). A COPY OF THE RESPONSE FORM IS SENT TO THE INITIATOR OF THE PERFORMANCE REPORT AND THE ORIGINAL OF BOTH ARE FILED BY SOFTWARE CONTROL.

TO OBTAIN ADDITIONAL FORMS AND DETAILED INSTRUCTIONS CONTACT KOREY WILLNAUER IN SOFTWARE CONTROL AT EXTENSION 2704, MAIL DROP 50.

TO SHOW HOW THIS PROCEDURE WILL WORK, I HAVE PREPARED A DETAILED EXAMPLE BASED ON OUR FIRST SUBMITTAL UNDER THE NEW SYSTEM. COPIES OF THE FORMS AND PRINTOUTS ARE ENCLOSED. THE PROCESS STARTED WHEN BOEN GO OBSERVED THAT THE 8048 CROSSASSEMBLER LIST OUTPUT REPORTED A PAGING ERROR WHENEVER THERE WAS AN ATTEMPT TO JUMP ACROSS A 2K BOUNDARY. HE FILED A PERFORMANCE REPORT AND INCLUDED A SAMPLE PROGRAM THAT ILLUSTRATES THE OBSERVED PROBLEM. THESE WERE REVIEWED BY PAUL SALAMON AND THEN WE DISCUSSED THE REPORT AND THE PROPOSED DISPOSITION.

AFTER STUDYING THE CROSSASSEMBLER, PAUL CONCLUDED THAT IT WOULD NOT BE POSSIBLE FOR THE CROSSASSEMBLER TO DETERMINE IF A PAGING ERROR HAD ACTUALLY OCCURRED. IN OTHER WORDS, BECAUSE OF THE PAGING ARCHITECTURE CHOSEN FOR THE 8048, IT IS NOT POSSIBLE TO AUTOMATICALLY DETECT PAGING ERRORS. THE CROSSASSEMBLER IS DESIGNER TO FLAG EVERY JUMP ACROSS A PAGE BOUNDARY. THE OBJECT CODE MAY BE CORRECT OR NOT DEPENDING ON IF THE PROGRAMMER MENTALLY KEEPS TRACK OF WHICH PAGE THE PROGRAM IS IN AND CORRECTLY IMPLEMENTS THE JUMPS.

OPTIONS TO TREAT THIS PROBLEM ARE:

1) REMOVE ALL REFERENCES TO PAGING ERRORS.
2) REDUCE MESSAGE TO WARNING OR INFORMATION STATUS.
3) LEAVE AS IS.

MY REACTION, FROM A QUALITY CONTROL STANDPOINT, IS THAT WE CANNOT BE RELEASING SOFTWARE THAT REPORTS ASSEMBLER ERRORS EVEN IF THE OBJECT CODE IS CORRECT AND THE DESIGNER HAS CORRECTLY ACCOUNTED FOR JUMPS ACROSS BOUNDARIES. TO A LESSER DEGREE, I ALSO OBJECT TO RELEASES THAT HAVE WARNINGS.

THE CURRENT STRUCTURE OF THE ASSEMBLER ALLOWS ONLY ERROR OR NON-ERROR STATUS AND WOULD HAVE TO BE MODIFIED TO INCLUDE WARNINGS OR INFORMATION FLAGS. THE ALTERNATIVE WE HAVE CHOSEN FOR THE PRESENT IS TO REMOVE ANY FLAG ON THE JUMP ACROSS BOUNDARIES. THIS ELIMINATES THE AUTOMATIC CODE CHECKING FEATURE FOR THE PROGRAMMER AND PUTS THE BURDEN ON HIM TO BE CAREFUL. THE BEST LONG-TERM SOLUTION IS TO HAVE ERROR WARNING AND INFORMATION STATUS REPORTED BUT WE DO NOT HAVE THE MANPOWER TO IMPLEMENT THESE CHANGES AT THIS TIME.

****
SUPPORT SOFTWARE PERFORMANCE REPORT
****

NAME: _____ DATE: _____

LOCATION: _____ PHONE: _____

PERFORMANCE OBSERVATION

DATE: _____ TIME: _____

CLASS:
____ PROBLEM/ERROR
____ SUGGESTED ENHANCEMENT
____ OTHER

PRIORITY:
____ HIGH
____ MEDIUM
____ LOW
____ NONE/SUGGESTION

USERNAME: _____
SUPPORT SOFTWARE FUNCTION NAME: _____

HOST CPU:
____ GOLD
____ SILVER

CAN PROBLEM BE DUPLICATED? _____
(ATTACH ANY SUPPORTING DOCUMENTS SUCH AS PRINTOUTS, VAX MESSAGES, OR COPIES OF APPLICABLE PAGES IN INSTRUCTION MANUALS.)

DESCRIBE PROBLEM, SUGGESTION, OR OBSERVATION AND MAKE RECOMMENDATIONS AS APPROPRIATE:_____

(FORWARD THIS FORM AND ALL SUPPORTING DOCUMENTS TO SCM., KOREY WILLNAUER, MAIL DROP 50, EXT.2704.)

FOR SCM USE ONLY: DATE RECEIVED: _____ REPORT NUMBER: _____

BY: _____

****
SUPPORT SOFTWARE PERFORMANCE RESPONSE
****

REPORT NUMBER: _____ DATE RECEIVED: _____

MAINTAINER: _____

PROPOSED DISPOSITION: _____

PERFORMANCE REPORT ENTERED INTO BUGLIST.LIS BY _____ (INITIALS) ON _____ (DATE).

MAINTAINER ACTION TAKEN: _____

MAINTAINER ACTION ENTERED INTO BUGFIX.LIS AND COPIES OF THIS RESPONSE SENT TO INITIATOR AND SCM ON _____ (DATE) BY _____ (MAINTAINER'S SIGNATURE)

****

FOR SCM USE ONLY: DATE LOGGED IN: _____ BY _____ (SIGNATURE)

REPORT AND RESPONSE LOG

REPORT NUMBER	REPORT DATE	REPORT INITIATOR	RESPONSE MAINTAINER	RESPONSE BUGLIST DATE	RESPONSE BUGFIX DATE

THIS IS THE COMMAND PROCEDURE USED BY THE SOFTWARE LIBRARIAN TO BE RUN ON THE SOURCE FILES TO CREATE THE LOAD FILE, THE ODD AND EVEN BYTE FILES (IF NEEDED), THE DOCUMENT FILES, AND THE ASCII-HEX FILES. THIS COMMAND PROCEDURE IS WRITTEN BY THE SOFTWARE ENGINEER PRIOR TO STAGE-3 RELEASE. ITS FILE NAME IS REQUIRED ON THE STAGE-3 RELEASE FORM. IT MUST CONTAIN THE EXPLICIT FILE NAME OR COMMAND LINE FOR EVERY SOURCE CODE FILE AND EVERY SUPPORT SOFTWARE PROGRAM.

TARGET SOFTWARE -- SEE PRODUCT SOFTWARE.

UNIT (LRU)

SECTION 8: HOW TO DESIGN YOUR COMMAND FILE

WHEN YOU RELEASE YOUR SOFTWARE INTO THE SYSTEM, YOU WILL BE ASKED BY SOFTWARE CONTROL TO PROVIDE BOTH A SOURCE FILE AND A COMMAND FILE SO THAT THE NECESSARY DOCUMENTATION FOR YOUR LOAD FILE RELEASE CAN BE GENERATED.

THE COMMAND FILE SHOULD BE ONE THAT YOU ACTUALLY USED TO FORMAT YOUR SOURCE FILE INTO DOWNLOADABLE CODE WHICH YOU THEN USED TO PROGRAM THE CHIPS WHICH YOU HAVE BEEN TESTING IN ENGINEERING UNITS.

THIS MEANS THAT THE COMMAND FILE AND THE CODE WHICH IT GENERATES WILL HAVE ALREADY BEEN TESTED AND PROVEN TO BE VALID.

THE SUPPORT SOFTWARE YOU USE WILL GENERALLY BE IMPLEMENTED IN THIS ORDER:

COMPILER (COMMAND INTERPRETER & CONVERSION)
INCLUDING: MACRO LIB / AS
RUN LIB /APPLICABLE

ASSEMBLER (COMMAND INTERPRETER & CONVERSION)

LINKER (COMMAND INTERPRETER & CONVERSION)

SEPARATOR

PARTITIONER

FORMATTER

YOU WILL ALWAYS NEED TO USE AN ASSEMBLER, A PARTITIONER, AND A FORMATTER IN YOUR COMMAND FILE. DEPENDING ON YOUR SOFTWARE, YOU MAY ALSO NEED TO USE THE OTHER SUPPORT SOFTWARE. SOFTWARE CONTROL SUGGESTS THE FOLLOWING FORMAT FOR A COMMAND FILE DESIGNED TO USE ALL OF THE SUPPORT SOFTWARE PACKAGES MENTIONED ON THE LOAD FILE RELEASE ORDER. YOU ARE AT PERFECT LIBERTY TO DESIGN YOUR OWN COMMAND FILE IF YOU WISH; HOWEVER, PLEASE BE SURE THAT IT INCLUDES THE INFORMATION PROVIDED BY THE EXAMPLE BELOW. (THE NUMBERS AT THE BEGINNING OF EACH LINE ARE FOR ILLUSTRATIVE PURPOSES ONLY AND WILL BE REFERRED TO IN THE NOTES BELOW; THEY DO NOT NEED TO BE IN THE COMMAND FILE.)

SAMPLE COMMAND FILE: SAMPLE.COM

```
1    $!  COMMAND PROCEDURE FOR CREATING LOAD AND DOCUMENT FILES FOR THE
2    $!  720-XXXX-XX LOAD FILE RELEASE.
3    $!
4    $!        NAME: (SOFTWARE PROJECT ENGINEER)     DATE: 17 JAN 1983
5    $!    MODIFIER:                                 DATE:
6    $!
7    $!  MAKE APPROPRIATE DIRECTORY ASSIGNMENTS.  THESE MAY HAVE TO
8    $!  BE MODIFIED WHEN RESTORING THE SUPPORT ENVIRONMENT AND TARGET
9    $!  SOURCE FILES FOR CHANGES.
10   $!
11   $ ASSIGN DRA1:[SOFTLIB.ARCHIVE.EXE] MICRO$DISK
12   $ ASSE*MBLE:==$MICRO$DISK:01ASMBLR
13   $!
14   $!  EXECUTE THE (8048) CROSS-ASSEMBLER.  VERSION CALLED BY
15   $!  01ASMBLR.EXE IS 01ASM8048.EXE.
16   $!
17   $ ASSEMBLE  KA594V01.I48/8048
18   $!
19   $!  EXECUTE THE PARTITIONER:
20   $!
21   $ RUN MICRO$DISK:02PART
22   KA594V01.LOA
23   2
24   722000501.DOC
25   400
26   0
27   722000601.DOC
28   800
29   400

30   $!
31   $!  EXECUTE THE FORMATTER:
```

```
32  $!
33  $ RUN MICRO$DISK:03FORMAT
34  722000501.DOC
35  I
36  $ RUN MICRO$DISK:03FORMAT
37  722000601.DOC
38  I
39  $ EXIT
```

THE FOLLOWING IS A LINE-BY-LINE COMMENTARY ON THE ABOCE COMMAND FILE.

LINES

1-2 THIS IS A UNIVERSAL HEADING WHICH CALLS OUT THE (720-) NUMBER OF THE LOAD FILE RELEASE.

4 THIS IS THE ORIGINAL DESIGNER OF THE COMMAND FILE AND THE DATE HE FIRST RAN IT IN ITS FINAL VERSION.

5 THIS IS THE NAME OF ANY PERSON WHO FINDS IT NECESSARY TO MODIFY THE COMMAND FILE (FOR INSTANCE FOR A SUBSEQUENT LOAD FILE RELEASE OCCURRING WHEN THE SOFTWARE IS REVISED). IF THE COMMAND FILE IS REVISED MORE THAN ONCE, THE MODIFIERS NAME AND THE DATE OF MODIFICATION SHOULD APPEAR FOR EACH CHANGE.

7-9 THIS IS A UNIVERSAL EXPLANATORY NOTE WHICH STATES THAT A SUBDIRECTORY IN THE SOFTWARE CONTROL FILES IS BEING SUBSTITUTED FOR THE SYSTEM MICROLIB DIRECTORY.

11 THIS LINE ACTUALLY MAKES THE NAME ASSIGNMENT EXPLAINED ABOVE.

12 THIS LINE STATES THE VERSION OF ASSEMBLER COMMAND INTERPRETER TO BE USED. IN THIS CASE IT IS 01ASMBLR. BE SURE TO CHECK IN [MICROLIB] FOR THE CURRENT VERSION AT THE TIME YOU RUN YOUR COMMAND FILE.

14-15 THIS IS EXPLANATORY MATERIAL STATING THE VERSION LEVEL OF THE CONVERSION ASSEMBLER WHICH WILL BE USED. NOTE THAT 3 ITEMS MAY VARY: THE DEVICE TYPE, THE COMMAND INTERPRETER USED, AND THE CONVERSION SOFTWARE USED.

17 THIS IS THE COMMAND TO ASSEMBLE YOUR SOURCE FILE. BOTH THE SOURCE FILE NAME AND THE DEVICE TYPE WILL NEED TO BE CHANGED FOR YOUR OWN COMMAND FILE. REMEMBER TOO THAT IF YOU ARE USING AN ASSMBLER WHICH IS NOT AN ABSOLUTE ASSEMBLER (WHICH AUTOMATICALLY PRODUCES RELOCATABLE RATHER THAN ABSOLUTE OBJECT CODE) BUT YOU WANT IT TO GENERATE AN ABSOLUTE OBJECT FILE, YOU NEED TO MAKE THE /ABS SPECIFICATION AT THE END OF THE COMMAND LINE.

18-20 THIS IS EXPLANATORY MATERIAL STATING THAT YOU WILL NOW RUN THE PARTITIONER.

21 THIS IS THE COMMAND TO RUN THE PARTITIONER. REMEMBER TO CHECK ON WHICH VERSION OF THE PARTITIONER IS CURRENT.

22 THIS IS THE INPUT TO THE PARTITIONER. NORMALLY IT WILL BE THE LOAD FILE JUST CREATED BY THE ABSOLUTE ASSEMBLER OR BY THE ASSEMBLER AND LINKER.

23 THIS IS THE NUMBER OF MEMORY PARTITIONS INTO WHICH THE PARTITIONER IS TO DIVIDE YOUR LOAD FILE. THE NUMBER OF MEMORY PARTITIONS WILL GENERALLY BE THE SAME AS THE NUMBER OF PROGRAMMABLE DEVICES IN THE PROCESSOR SYSTEM.

24 THIS IS THE NUMBER OF THE FIRST BINARY IMAGE FILE (CORRESPONDING TO THE FIRST PROGRAMMED DEVICE) TO BE GENERATED. IT WILL ALWAYS BEGIN WITH '722' AND END WITH '.DOC' . IT IS THE 722- NUMBER ASSIGNED IN STAGE 3, WITHOUT THE DASHES.

25 THIS IS THE SIZE IN HEX OF THE FIRST PARTITION. NORMALLY THIS WILL BE THE MEMORY SIZE OF THE FIRST CHIP.

26 THIS IS THE ADDRESS IN THE LOAD FILE WHERE THE CODE FOR THE FIRST CHIP BEGINS. UNLESS YOU ARE USING AN OFFSET, THIS WILL NORMALLY BE 0.

27 THIS IS THE NUMBER OF THE SECOND BINARY IMAGE FILE (CORRESPONDING TO THE SECOND PROGRAMMED DEVICE) TO BE GENERATED. AGAIN, THIS NUMBER WILL HAVE BEEN ASSIGNED ON THE STAGE 3 NUMBER ASSIGNMENT FORM.

28 THIS IS THE SIZE IN HEX OF THE SECOND PARTITION. NORMALLY THIS WILL BE THE MEMORY SIZE OF THE SECOND CHIP.

29 THIS IS THE ADDRESS IN THE LOAD FILE WHERE THE CODE FOR THE FIRST CHIP BEGINS. UNLESS YOU ARE USING AN OFFSET, THIS WILL NORMALLY BE 0.

30-32 THIS IS EXPLANATORY MATERIAL STATING THAT YOU WILL NOW RUN THE FORMATTER.

33 THIS IS THE COMMAND TO RUN THE FORMATTER. REMEMBER TO CHECK ON WHICH VERSION OF THE FORMATTER IS CURRENT.

34 THIS IS THE INPUT FILE TO THE FORMATTER. THIS WILL BE THE .DOC BINARY IMAGE FILE CREATED ABOVE.

35 THIS IS THE STRUCTURE TYPE OF THE PROM PROGRAMMER YOU ARE USING. IT WILL BE EITHER I(NTEL), M(OTOROLA), OR T(EKTRONIX).

36 THIS IS THE REPEAT COMMAND TO RUN THE FORMATTER IN CASES WHERE YOU ARE DEALING WITH MORE THAN ONE CHIP. (THE FORMATTER MUST BE RUN FOR EACH.)

37 THIS IS THE INPUT FILE FOR THIS SESSION OF FORMATTING. HERE IT IS THE BINARY IMAGE FILE NUMBER FOR THE SECOND PROGRAMMED CHIP.

38 THIS IS AGAIN THE STRUCTURE TYPE OF THE PROM PROGRAMMER YOU ARE

USING.
39 THIS IS THE COMMAND WHICH STATES THAT THIS IS THE END OF THE COMMAND FILE AND THAT CONTROL SHOULD BE RETURNED TO THE USER WITH THE $ PROMPT.

** 10.0 COMMAND FILE FOR EXECUTION OF SUPPORT SOFTWARE

** 10.1 INTRODUCTION

WHEN TARGET SOFTWARE IS RELEASED TO MANUFACTURING, GOOD CONFIGURATION MANAGEMENT PRACTICES REQUIRE THAT THE SOURCE CODE AND NECESSARY SUPPORT SOFTWARE BE CAREFULLY DOCUMENTED AND ARCHIVED. A TECHNIQUE IS NEEDED FOR RECORDING PRECISE VAX FILE NAMES FOR TARGET SOURCE CODE AND SUPPORT SOFTWARE. THE GOAL IS TO BE ABLE TO RECREATE THE VARIOUS OBJECT, LOAD, AND DOCUMENT FILES WHENEVER MAINTANENCE OR REVISION REQUIREMENTS ARIZE. REVISION HISTORY FOR BOTH TARGET AND SUPPORT SOFTWARE MUST BE MAINTAINED FOR THE LIFE OF THE PRODUCT.

TO OBTAIN THIS PRECISE RECORD, THE OBJECT, LOAD, AND DOCUMENT FILES ARE CREATED ON VAX BY RUNNING A DOCUMENTED SUPPORT SOFTWARE COMMAND PROCEDURE (SSCP) THAT REFERENCES UNIQUELY-NAMED SOURCE CODE FILES AND SUPPORT SOFTWARE. IN ADDITION, ALL FILES ASSOCIATED WITH A RELEASE ARE ARCHIVED ON MAGNETIC MEDIA FOR FUTURE USE.

THE SUPPORT SOFTWARE COMMAND PROCEDURE, AND THE SUPPORT AND TARGET SOFTWARE FILES SPECIFIED BY IT, ARE USED BY THE SOFTWARE LIBRARIAN TO CREATE THE LOAD FILE, THE ODD AND EVEN BYTE FILES (IF NEEDED), THE DOCUMENT FILES, AND THE ASCII-HEX FILES. THE SSCP AND TARGET SOURCE FILES ARE PROVIDED TO SOFTWARE CONTROL, THROUGH THE USE OF THE [EXCHANGE] DIRECTORY, AT THE TIME OF STAGE-3 RELEASE.

THE SSCP MUST USE THE EXPLICIT FILE NAMES AND COMMAND LINES FOR ALL SOURCE CODE AND SUPPORT SOFTWARE FILES. NO WILD CARDS OR DEFAULT SPECIFICATIONS CAN BE USED. ALL SUPPORT SOFTWARE USED FOR RELEASE WILL BE IN A DIRECTORY NAMED [MICROTEC.ARCHIVE]. THE FOLLOWING IS A LIST OF POSSIBLE SUPPORT SOFTWARE MODULES:

COMMAND INTERPRETERS
  COMPILER
  ASSEMBLER
  LINKER
CONVERSION
  COMPILER
  MACRO LIBRARY
  RUN LIBRARY
  ASSEMBLER
  LINKER
OUTPUT
  SEPARATOR
  PARTITIONER
  FORMATTER

THE EXACT FILE NAMES OF THE LATEST VERSIONS OF SUPPORT SOFTWARE CAN BE OBTAINED FROM SUPPORT SOFTWARE PERSONNEL.

** 10.2 EXAMPLE I.

THE FOLLOWING IS AN EXAMPLE OF A SUPPORT SOFTWARE COMMAND FILE USED TO RELEASE A LOAD FILE THAT IS PROGRAMMED INTO TWO MEMORY DEVICES.

```
$!     THE FIRST COMMAND LINE RUNS THE COMMAND INTERPRETER FOR THE
$!     I8048 ABSOLUTE CROSS-ASSEMBLER.  THE FILE NAME FOR THE
$!     ASSEMBLER VERSION THAT IS ACTUALLY EXECUTED IS:
$!     [MICROTEC.ARCHIVE]00ASM8048.EXE.
$
$ @[MICROTEC.ARCHIVE]00ASM.COM KMRCC.I48/8048/NOLIST
$
$!     THE SOURCE FILE NAME IS KMRCC.I48 AND THE NOLIST QUALIFIER
$!     IS USED SO THAT THE LISTING FILE WILL NOT BE PRODUCED.
$!
$!     THE OUTPUT FROM THE ABSOLUTE CROSS-ASSEMBLER IS A LOAD FILE
$!     NAMED KMRCC.LOA.  THE LOAD FILE IS PROCESSED BY THE PARTITIONER
$!     TO PARTITION A LARGE LOAD FILE OVER SEVERAL MEMORY DEVICES,
$!     IF NECESSARY, AND TO CREATE A BINARY IMAGE DOCUMENT FILE FOR
$!     EACH DEVICE.  THE COMMAND LINE BELOW RUNS THE PARTITIONER AND
$!     THE SEVERAL LINES FOLLOWING IT ARE RESPONSES TO PARTITIONER
$!     PROMPTS FOR INFORMATION.
$
$       RUN [MICROTEC.ARCHIVE]00PART.EXE
KMRCC.LOA
2
722000500.DOC
400
0
```

```
722000600.DOC
800
400
$
$!     THE FIRST LINE AFTER THE RUN COMMAND IS THE FILE NAME OF THE
$!     LOAD FILE TO BE PROCESSED.  IT DOES NOT REQUIRE A DIRECTORY
$!     SPECIFICATION BECAUSE IT IS PROCESSED IN A SPECIFIC DIRECTORY
$!     MANAGED BY THE SOFTWARE CONTROL LIBRARIAN.  THE NEXT LINE IS
$!     THE TOTAL NUMBER OF MEMORY DEVICES THAT THE LOAD FILE WILL BE
$!     PARTITIONED INTO.  THE NEXT GROUP OF THREE LINES APPLY TO THE
$!     FIRST MEMORY DEVICE.  THE GENERAL PATTERN OF THREE LINES WILL
$!     BE REQUIRED FOR EVERY MEMORY DEVICE.  THE FIRST LINE IN EACH
$!     GROUP OF THREE SPECIFIES THE DOCUMENT FILE NAME FOR THE BINARY IMAGE
$!     FILE.  THE SECOND LINE SPECIFIES THE SIZE OF THE MEMORY DEVICE
$!     (TOTAL NUMBER OF MEMORY LOCATIONS) IN HEX.  THE THIRD LINE
$!     SPECIFIES THE STARTING ADDRESS IN THE MAIN LOAD FILE THAT
$!     CORRESPONDS TO THE FIRST MEMORY ADDRESS (0000H) IN THE DEVICE.
$!     IN THIS EXAMPLE, A 1K MEMORY DEVICE IS ASSIGNED THE FIRST
$!     400H LINES OF CODE IN THE MAIN LOAD FILE NAMED KMRCC.LOA STARTING
$!     AT ADDRESS 0000H.  THE RESULTING BINARY IMAGE DOCUMENT FILE IS
$!     NAMED 722000500.DOC.  THEN, A 2K MEMORY DEVICE IS ASSIGNED THE
$!     REMAINING CODE IN THE SAME LOAD FILE STARTING AT ADDRESS 0400H.
$!     THE PARTITIONER WILL AUTOMATICALLY FILL EVERY UNEXECUTED MEMORY
$!     LOCATION WITH FF.  THE BINARY IMAGE DOCUMENT FILE NAME IS
$!     722000600.DOC.
$!
$!     THE BINARY IMAGE FILE IS NOT LOADABLE BECAUSE IS IS NOT IN
$!     ASCII-HEX FORMAT.  TO CONVERT THE PARTITIONED CODE INTO A
$!     DOWNLOADABLE FORMAT REQUIRES THE FORMATTER.  SINCE THE FORMATTER
$!     OPERATES DIRECTLY ON THE DOCUMENT FILE, IT PROVIDES DIRECT
$!     VERIFICATION THAT THE BINARY IMAGE DOCUMENT FILE IS CORRECT.
$!
$!     THE FOLLOWING COMMAND LINES RUN THE FORMATTER AND PROVIDE
$!     INFORMATION IN RESPONSE TO THE VARIOUS FORMATTER PROMPTS.
$
$ RUN [MICROTEC.ARCHIVE]00FORMAT.EXE
722000500.DOC
I
$ RUN [MICROTEC.ARCHIVE]00FORMAT.EXE
722000600.DOC
I
$
$!     THE INFORMATION REQUIRED BY THE PROMPTS ARE THE INPUT DOCUMENT
$!     FILE NAMES TO BE FORMATTED INTO ASCII-HEX AND THE TYPE OF FORMAT
$!     DESIRED (INTEL, MOTOROLA, TEKHEX).  IN THIS EXAMPLE, THE
$!     ASCII-HEX FILE FOR THE FIRST DEVICE IS TO BE PRODUCED IN INTEL
$!     FORMAT FROM THE DOCUMENT FILE NAMED 722000500.DOC.
$
$       EXIT
DATE: 6 OCT 1982
NAME: STEVE RUSSELL

THIS IS THE START OF AN EXAMPLE TO THE NEW COMMAND FILE FORMAT NEEDED FOR
SOFTWARE RELEASES.

1. THE DESIGNER WRITES THE COMMAND PROCEDURE.  HIS PROCEDURE MUST
   EXPLICITELY DEFINE THE FOREIGN COMMAND VERSION.  HE MUST USE LOGICAL
   NAME ASSIGNMENTS FOR DEVICES AND DIRECTORIES.

2. THE LIBRARIAN RUNS THE EXACT SAME COMMAND PROCEDURE BUT MUST DEFINE
   THE LOGICAL NAMES TO ACCESS THE PROPER DISK AND DIRECTORY.  THIS CAN
   BE DONE IN THE LOGIN PROCEDURE SO IT WOULD BE TRANSPARENT TO THE
   DAILY WORK.


EXAMPLE:


$ ASSEMBLE:==$MICRO$DISK:00ASMBLR.EXE
$ ASSEMBLE/NOLIST/8048  KA120V01.I48
$ RUN MICRO$DISK:01PART.EXE
KA120V01.LOA
720000000
$ RUN MICRO$DISK:01FORMAT.EXE
$ EXIT
```

THE FOLLOWING IS AN EXAMPLE OF HOW YOUR SOURCE CODE CAN BE HEADED

```
;*****************************************************************************
;
;       PROGRAMMER:                 JOE ENGINEER
;       SYSTEM NAME:                KZZ 999 WEATHER BALLOON
;       UNIT NAME:                  KXX 123 RANGE FINDER
;       PROCESSOR SYSTEM NAME:      KXX 123 ANALOG PROGRAMMER
;       VAX FILE NAME:              [SOFTLIB.720123456]KXX123.I48
;       DATE:                       24 MAY 1983
;
;*****************************************************************************
;
;       DESCRIPTION:                INTEL ASSEMBLY SOURCE CODE FOR THE
;                                   KXX 123 ANALOG PROGRAMMER.
;
;*****************************************************************************
;
;       REVISION HISTORY:
;
;*****************************************************************************
;
;       TITLE:                      KXX 123 ANALOG PROGRAMMER
;
;*****************************************************************************
```

THE FOLLOWING IS AN EXAMPLE OF HOW YOUR COMMAND FILE CAN BE HEADED

```
$!      COMMAND PROCEDURE FOR CREATING LOAD AND DOCUMENT FILES FOR
$!      THE 720-XXXX-XX LOAD FILE RELEASE.
$!
$!      NAME:   (SOFTWARE PROJECT ENGINEER)     DATE:  24 MAY 1983
$!      MODIFIER:                               DATE:
$!
$!      MAKE APPROPRIATE DIRECTORY ASSIGNMENTS.  THESE MAY HAVE TO BE
$!      BE MODIFIED WHEN RESTORING THE SUPPORT ENVIRONMENT AND TARGET
$!      SOURCE FILES FOR CHANGES.
$!
```

APPENDIX 14. MCAIR QA PLAN

3.3 SOFTWARE CONFIGURATION CONTROL

CONFIGURATION CONTROL PROCEDURES AT KING RADIO CORPORATION ARE DESIGNED TO ACCOMPLISH THE FOLLOWING:

1) ACCURATE IDENTIFICATION OF EVERY ITEM OF BOTH PRODUCT AND SUPPORT SOFTWARE USED TO PRODUCE THE MACHINE CODE FOR A SPECIFIED PRODUCT RELEASE VERSION.
2) COMPLETE DOCUMENTATION OF THE SOFTWARE DESIGN.
3) DOCUMENTATION OF EACH SOFTWARE RELEASE VERSION.
4) ARCHIVING OF ALL PRODUCT SOFTWARE AND SUPPORT SOFTWARE.
5) MAINTANENCE OF THE ARCHIVE OVER THE ENTIRE PRODUCT LIFETIME.
6) MONITORING OF EACH STEP OF THE RELEASE TO INSURE ACCURACY.

CONFIGURATION CONTROL OF SOFTWARE IS ACCOMPLISHED THROUGH THE USE OF REVISION CONTROLLED DOCUMENTS IN A MANNER ANALOGOUS TO THE CONTROL PROCEDURES USED FOR HARDWARE.

CONTROLLED SOFTWARE DOCUMENTS ARE:

1) SOFTWARE REQUIREMENTS DOCUMENT
2) LRU SOFTWARE DOCUMENT
3) UNIT SOFTWARE CID
4) PROGRAMMING LITERATURE BIBLIOGRAPHY
5) SOURCE LISTING DOCUMENT
6) LOAD FILE RELEASE DOCUMENT
7) DESIGN DESCRIPTION DOCUMENT
8) MACHINE CODE DOCUMENT

A SOFTWARE CONFIGURATION CONTROL ORGANIZATION IS RESPONSIBLE FOR ALL QA, DOCUMENTATION, AND CONFIGURATION CONTROL ACTIVITIES FOR NEW SOFTWARE RELEASES OR REVISIONS TO EXISTING SOFTWARE. NEW RELEASES AND REVISIONS ARE HANDLED IN A VERY SIMILIAR, BUT NOT IDENTICAL, MANNER. ANY CHANGES TO THE DOCUMENTATION OR CONFIGURATION MUST BE APPROVED BY SOFTWARE CONFIGURATION CONTROL.

MODIFICATIONS TO EXISTING SOFTWARE ARE CONTROLLED BY THE ENGINEERING CHANGE ORDER (ECO) SYSTEM. IN ADDITION TO THE NORMAL PROCEDURES ESTABLISHED FOR HARDWARE, A SOFTWARE ECO MUST ALSO BE PROCESSED AND APPROVED BY SOFTWARE CONFIGURATION CONTROL.

ALL PROCEDURES ASSOCIATED WITH THE DEVELOPMENT, RELEASE, AND MAINTANENCE OF PRODUCT SOFTWARE ARE SPECIFIED IN THE SOFTWARE METHODOLOGY DOCUMENT.